سلسلة نظام MSX ٢

البرمجة بلغة الآلة لنظام MSX

ج. ب. ريدلي

ترجمة واعداد:
محمد المدني

البرمجة بلغة الآلة

لنظام الـ MSX

ج . ب . ريدلي

ترجمة واعداد

محمد المدني

* البرمجة بلغة الآلة لنظام الـ MSX
* تأليف ج . ب . ريدلي
* ترجمة واعداد: محمد المدني
* الطبعة الأولى ٢٠٠٠/ ٨/ ١٩٨٨

* دار الحضارة للنشر والتوزيع
* دمشق، ص . ب ١١٢٨٢ ، هاتف ٢١٢٦٨٥

* تنفيذ: الاهالي للطباعة والنشر والتوزيع
* دمشق، ص . ب ٩٥٠٣، هاتف ٤٢٠٢٩٩، تلكس ٤١٢٤١٦

مقدمة

Introduction

لقد كتب هذا الكتاب كمقدمة لكتابة برامج بلغة الآلة وبرامج فرعية مستخدما لغة التجميع لنظام الـ MSX المنتشر في الحواسب المنزلية .

قبل سنين عديدة كانت لغة الآلة هي لغة المبرمج الأولى ، ولكن مع شيوع الحواسب المنزلية أصبحت لغة البيزيك (Basic) أكثر شيوعاً ، ومعظم مستخدمي الحواسب الصغيرة تطوروا معها . وبقيت لغة الآلة نوعاً ما كمساحة رمادية اللون التي يراها معظمنا في قائمة البرنامج سلسلة من الأرقام في تعليمة الـ DATA محجوزة في منطقة الذاكرة العليا والتي تستدعى عندئذٍ بأمر (USR) .

وتترك بغير إشارة لما حدث :

برامج لغة الآلة ، تعمل بسرعة أكبر من تلك التي كتبت بلغة البيزيك Basic وذلك أحد أسباب تضمين برامج لغة البيزيك ببرامج فرعية بلغة الآلة لتقوم بإنجازها بسرعة أكبر . أو يمكن أن تستخدم لتعديل برامج لغة البيزيك Basic لتقوم بتنفيذ أعمال لا يمكن أن تقوم بها بالحالة العادية .

نرجو من هذا الكتاب أن يجعل لغة الآلة أوضح وأكثر قابلية للفهم والاستيعاب لعدد كبير من المستخدمين الذين لا يحتاجون إلى درجة عالية من الإدراك لما يحدث والاحتفاظ برطانة الكومبيوتر إلى الحدود الدنيا .

حظاً سعيداً .

الفصل الأول
لغة الآلة من البيزيك
Machine Code From Basic

إن لغـة (البيـزيـك) عموماً هي أبسط طريقة لكتابة البرامج، إنها سهلة في التتبع وفي اكتشاف الأخطاء وتعديل أسطر البرنامج بتسهيلات التعديل الموجودة، إذاً لماذا تستعمل لغة الآلة؟

الهدف الرئيسي يجب أن يكون السرعة في التنفيذ، ليس كقاعدة فقط في برامج لألعـاب مثـل غزو الفضـاء أو مثل تلك الألعاب التي لن تكون أفضل ان كتبت بلغة البيـزيـك BASIC ولكن سوف نشـاهد في هذا الكتاب تطبيقات أكثر جدية، من أجل ادراك فكـرة السـرعـة لبرنـامـج قد كتب بلغة التجميع سوف نقارن زمـن التنفيذ مع برنامج مماثل كتب بلغة البيزيك BASIC .

برنامج :

```
10 SCREEN0:KEYOFF
20 WIDTH40:CLS
30 TIME=0
40 FOR X = 0 TO 959
50 PRINT"B";
60 NEXT
70 PRINT TIME
```

اضغـط الآن المفتـاح «F5» أو أدخـل «RUN» متبـوعـة بمفتـاح نهايـة السطـر «RETURN» سوف ترى أن نظام MSX قد استغـرق 151 دورة زمنيـة (أو 3.02 ثانية إذا عدل السطر 70 إلى TIME/50) حتى تُملأ الشاشة بالحرف «B»

هناك طريقة أخرى لإظهار الأحرف وهي باستخدام تعليمة (VPOKE) مباشرة بتحديد الموقع على الشاشة. في الشاشة ذات (الموديل) صفر (SCREEN 0) إن الموقع العلوي اليساري (الزاوية اليسارية من الشاشة) لهذه الشاشة ذات الأربعين (40) عاموداً يكون صفراً (000 hex) في (نظام الستة عشرة)، دعنا نبدل البرنامج السابق وبدلاً من طباعة حرفٍ تلو الآخر على الشاشة باستخدام تعليمة البيزيك (PRINT) سوف نطبع الأحرف مباشرة في منطقة الشاشة في الذاكرة مستخدمين تعليمة (VPOKE).

أضف السطر 15 إلى البرنامج السابق:

15 Z = & H0

وبدّل الأسطر التالية لتصبح:

```
50 VPOKEZ+X,66
70 LOCATE,23:PRINT TIME
```

مرة أخرى أعد تشغيل البرنامج.

لقد كان الوقت 255، إذاً التخزين المباشر في منطقة الشاشة ليس أسرع.

لاحظ أن النقطة المضيئة تبقى في نفس الموقع حين نُفِذّت عملية التخزين (VPOKE)، لذلك السطر 70 يقوم بإعادة تَوَضع النقطة المضيئة على السطر 23 من الشاشة بوساطة تعليمة التوضع (LOCATE)

إذا وجدت إشارة «OK» قد ظهرت على الشاشة أدخل عندئذٍ وبالطريقة المباشرة (الموديل) «WIDTH 37» واضغط مفتاح نهاية السطر «RETURN» لتمحى الشاشة ويرجع عدد الأعمدة 37 كما هو الحال عند تشغيل نظام الـ MSX لأول مرة.

إذا اعتبرنا هذا البرنامج ليس مفيداً كثيراً لكنه فعَّال تماماً كمثال إذا أعيد كتابته بلغة الآلة وتم إستدعاؤه بالأمر A = USR (0) في لغة البيزيك فإن الزيادة الدراماتيكية في السرعة سوف تكون واضحة فوراً.

VPOKE تعليمة موجودة في نظام MSX وتعني تخزين رقم ما مباشر في منطقة ذاكرة الشاشة. VIDEO RAM POKE

البرنامج ١ ـ العنونة المباشرة للشاشة بلغة البيزيك:

أوامر لغة التجميع تستخدم:

LD HL, nnnn LD BC, nnnn LD A,nn CALL nnnn RET

«هذه الأوامر شرحت بالتفصيل في الفصل الثاني».

أدخل «NEW» و «RETURN» وأكتب البرنامج التالي:

```
10 CLEAR 200,&H9FFF
20 FOR X = &HA000 TO &HA00B
30 READ A:POKE X,A:NEXT
40 DATA 62,66,33,0,0,1,192,3,205,86,0,201
```

اضغط المفتاح «F5» لتشغيل البرنامج.

الشاشة مباشرة هذه المرة أظهرت إشارة «OK».

لا أحد يستطيع أن يفكر بأن شيئاً قد حدث. لكنه حدث، فقد أخذ البرنامج الفرعي المكتوب بلغة الآلة مكانه في الذاكرة، يبدأ من الموقع (A 000 hex) (في النظام الستة عشر) (عشري 40960)، الذي سوف يطبع كامل الشاشة بالحرف «B» بأجزاء من الوقت السابق عندما كتب البرنامج بلغة البيزيك مستخدماً التعليمة PRINT أو VPOKE.

أدخل «NEW» مع «RETURN» وأكتب البرنامج التالي:

```
10 DEF USR=&HA000:SCREEN0
20 TIME=0
30 A=USR(0)
40 LOCATE,23:PRINT TIME
```

ـ نفذ هذا البرنامج:

إن سرعة تنفيذ هذا البرنامج مذهلة وخيالية والوقت قد سجل على الشاشة 1 أو 2.

ولسوف تشاهد في الفصل القادم أن لغة التجميع مؤلفة من عدة سجلات (تشحن) بعناوين وقيم. وبإمكانك التأكد من هذه الرموز في الملحق الموجود في آخر هذا الكتاب.

وإذا قمنا بتفكيك المعطيات التي توضّعت من العنوان A000 إلى A00B فإنها سوف تكون على النحو التالي:

```
1    A000    3E 42        LD A,42H
2    A002    21 00 00     LD HL,0000H
3    A005    01 C0 03     LD BC,03C0H
4    A008    CD 56 00     CALL 0056H
5    A00B    C9           RET
```

لقد خزنا المعطيات في الذاكرة مبتدئين من العنوان (A000 hex) بالنظام الستة عشر الذي إذا حولناه إلى النظام العشري أعطانا 40960 «تأكد من الملحق إذا لم تكن متأكداً من ذلك».

إن أول رقمين في سلسلة المعطيات (DATA) كانا (62,66) عشري. 62 بعد تحويلها إلى (النظام الستة عشر) تصبح 3E والتي تعني أننا نريد (شحن) السجل (A) بالقيمة التي في (البايت) (BYTE) التالية والتي في هذه الحالة كانت 66 (42 hex) ستة عشر في السطر الأول من البرنامج الأخير في الصفحة السابقة.

ثلاثة الـ(بايتات) (BYTES) التاليات والتي كانت 33,0,0 والتي هي (21,00,00 hex) في (نظام الستة عشر).

(21 hex) تشير إلى (شحن) السجل المزدوج (HL) بالبايتين (BYTES) المتتاليتين وبشكل معكوساً، العنوان ذو المرتبة الأدنى أولاً، في مثالنا هذا نريد أن (نشحن) السجل (HL) بعنوان الزاوية العليا اليسارية من الشاشة ذات (الموديل) صفر (Screen 0) (منطقة ذاكرة الشاشة) والتي تكون (0000 hex) (ستة عشر).

لذلك في هذه الحالة بالتحديد عملية العكس لا ترينا أي شيء لأن العنوان صفر في هذه الحالة ولكن السطر التالي سوف يوضح الفكرة.

بعد البايتات (BYTES) الثلاث الأخيرة نرى البايتات الثلاث المتتالية 1,192,3 ، الرقم 1 يشير إلى (شحن السجل (BC) بالبايتين (BYTES) المتتاليتين، بحيث البايت (BYTE) الأدنى أولاً ثم البايت (BYTE) الأعلى. في هذه المرحلة المراد (شحن) السجل (BC) بمقدار البايتيات (BYTES) التي نريد طباعتها على الشاشة.

الشاشة (مويل) صفر (screen0 تحتوي على السعة العظمى مقدرة بـ 960 موقع، 24 سطراً بـ 40 عاموداً، لذلك فإن 960 عشري تساوي إلى (03c0 hex) (ستة عشر والتي بالشكل المعكوس تصبح (C003) «السطر الثالث من البرنامج الأخير».

يأتينا بعد ذلك 205,86,0 . حيث 205 تساوي (CD hex) (ستة عشر) التي تقوم باستدعاء عنوان البايتين (BYTES) التالي والتي تكون بشكل معكوس.

ان عنوان البرنامج الفرعي الذي نريد استدعاءه هو (0056 hex) لذلك إذا عكسنا هذا العنوان وحولناه إلى العشري يصبح 86,0 .

ملاحظة:

ان منطقة ال ROM من الذاكرة تحوي عادة على عدد من البرامج الفرعية التي يمكن استدعاؤها لتقوم بتنفيذ أوامر مختلفة. الموقع 0056hex يحوي التعليمات للقفز إلى البرنامج الفرعي الذي يقوم بملء منطقة ذاكرة الشاشة بالحرف المخزن في السجل (A).

وعلى كل حال قبل استدعاء البرنامج الفرعي يجب أن نكون متأكدين من أن:

السجل HL يحوي على عنوان البداية، السجل BC يحوي على عدد البايتات (BYETS) والسجل A على المعطيات. وهذا ما يقوم به برنامجنا من السطر 1 إلى السطر 3.

الرقم الأخير في سطر المعطيات كان 201 الذي يساوي (C9 hex) ، وهذه هي تعليمة RET والتي تعني الرجوع إلى برنامج البيزيك، تماماً كما تفعله تعليمة (RETURN) عندما تستخدم مع ال GOSUB في البيزيك. تذكر أننا استدعينا هذا البرنامج الفرعي بوساطة الأمر USR(0) والذي يقوم باستدعاء التعليمات كما هو الحال في لغة البيزيك GOSUB وللخروج من البرنامج الفرعي أدخلنا الأمر RET للرجوع إلى برنامج البيزيك.

الأمر USR يمكن أن يحوي بين القوسين ( ) عدد صحيح، صفر، أو متغير أحادي الدقة أو متغير ذو دقة مضاعفة للدخول إلى برنامج لغة الآلة واستخدامه. ولقد شرح بالتفصيل في الفصل الرابع من هذا الكتاب.

ولكن من أجل مثالنا لم يكن هناك أي معطيات يراد معالجتها سوى استدعاء بسيط، ولهذا استخدم صفر بين القوسين (0).

والآن إن هذا البرنامج الفرعي قد نفذ بأجزاء من الوقت الذي استغرقه عندما كتب بلغة البيزيك لأن لغة البيزيك ليست سريعة (للبرمجة). وهناك الكثير أيضاً يرغبون بانتاج اخراجات بسيطة كتلك. ولكن أيضاً عملية الترجمة والتحويل من القيم العشرية إلى القيم (الستة عشر) وترميز العمليات وترجمتها إلى لغة التجميع هي عملية معقدة وليست بسيطة.

في أمثلتنا اللاحقة لبرامج لغة الآلة سوف نستخدم برنامج التجميع ASSEMBLER الذي يدعى «ZEN-» وهو نظام (البرمجة بلغة التجميع للمعالج Z 80 للميكروكومبيوتر) الذي يعمل على نظام MSX، والذي يتضمن برنامج تحويل لغة الآلة إلى لغة التجميع «Disassembler».

إن برنامج التجميع Assembler سوف يقوم بمعظم الأعمال الصعبة ويقدم لك الإخراجات كما رأينا سابقاً، وإضافة إلى ذلك فإن ادخال لغة التجميع قد صنعت بطريقة سهلة كلعبة الأطفال، وتسمح بإدخال رمز العملية Opcodes، والمعامل الذي تجري عليه العملية Operands كما يلي: LD BC, 03C0 H مباشرة.

بعد إدخال القائمة التي اختيرت، يقوم البرنامج المجمع عندئذ، بترجمة التعليمات إلى لغة الآلة (أوتوماتيكياً) ونسخة الإخراج الناتجة عن هذه العملية تُعرف بالترميز بلغة الآلة OBJECT وهذا الجزء من العمل يعني ببساطة تجميع لغة الآلة وجعلها مخزنة وجاهزة كسجل على شريط مغناطيسي (لشحنها) في المستقبل.

في الواقع من المستحيل كتابة برامج لغة الآلة مهما كانت حجمها بدون برنامج التجميع، فإنه سوف يقوم بالتقاط الجمل الخطأ كما هو الحال في لغة البيزك عندما يعطي رسائل تنبيه عن الخطأ. ويسمح برنامج التجميع بتشغيل هذه البرامج وايقافها عند نقاط معينة لتقوم باختبار حالة بعض السجلات.. الخ.

ويعد هذا شيئاً مهماً جداً في عملية تشغيل البرامج بشكل سريع وإنه من الصعب القيام بهذه الاختبارات بدون هذه التسهيلات.

البرنامج الثاني ـ تخزين الشاشات:

تعليمة جديدة من لغة التجميع استخدمت:

LD DE, nnnn

هناك برنامجان فرعيان في منطقة ال ROM يسمحان بنسخ منطقة الشاشة إلى قسم آخر من الذاكرة لتخزن وتستدعى عند الحاجة.

يمكن أن يكون هذا (الروتين) ضمن قائمة الاختيارات لبرنامج ما حيث يعطي الخيار للمستثمر بإستدعاء الشاشة مرة أخرى.

ويمكن تخزين إظهارات الشاشة كاملة في منطقة ما من الذاكرة RAM وعند الحاجة لها يمكن أن تسترجع بالأمر A = USR (0) مباشرة حيث تنقل تلك الكتلة من الذاكرة إلى الشاشة.

أدخل NEW و RETURN واكتب البرنامج التالي:

```
10 CLEAR200,&HDFFF
20 DEF USR0=&HF000:DEF USR1=&HF010
30 FOR X = &HF000 TO &HF00C
40 READ A:POKE X,A:NEXT
50 DATA 33,0,0,17,0,224,1,192,3,195,89,0,201
60 FOR X = &HF010 TO &HF01C
70 READ A:POKE X,A:NEXT
80 DATA 33,0,224,17,0,0,1,192,3,195,92,0,201
```

اضغط المفتاح «F5» أو «RUN» مع «RETURN».

مرة اخرى اشارة ال OK ظهرت مباشرة ولدينا الآن البرنامج الفرعي الذي يخزن الشاشة في الذاكرة. لقد زودنا الآن بنسخة عن الشاشة المعروضة وإذا لم يكن هناك شيء موجود على الشاشة، فضع أي شيء تريد.

أدخل وبالطريقة المباشرة «دون رقم سطر» A = USR (0) و «RETURN» . فإن إشارة الـ «OK» سوف تظهر من فورها وإن منطقة الإظهار «منطقة الشاشة» قد نسخت في الذاكرة من الموقع E 000 إلى الموقع E 3 BF hex (ستة عشر) .
إن الشاشة لم تختف لكن نسخة ثانية عنها وضعت في منطقة أخرى من الذاكرة .

وإذا نفذ شخص برنامجاً فإن من المحتمل أن تكون الشاشة الآن نظيفة والبرنامج ما زال موجوداً عندما نحتاجه لإسترجاع أي شاشة سابقة .
الآن نظف الشاشة بضغط المفتاحين «SHIFT» و «HOME» وحتى نثبت إمكانية البرنامج السابق حاول أن تكتب أي شيء على الشاشة وليس هناك أي مشكلة إذا ظهرت رسالة الخطأ «Syntaxerron» فقط أدخل ما تريد على الشاشة .
أدخل وبالطريقة المباشرة A = USR 1(0) مع ضغط «ENTER»
سوف تظهر مباشرة الشاشة السابقة التي كنا قد حفظناها عندما أدخلنا A = USR (0) .

نستطيع أن نحفظ أكثر من شاشة في الذاكرة بتزويد البرنامج بالأماكن المختلفة التي سوف توضع الشاشات بها . فالشاشة ذات الموديل) صفر (Screen 0) تحتوي حوالي 960 بايت (BYETS) فقط سوف نحتاج إلى تعديل البرنامج السابق من أجل مناطق تخزين مختلفة في الذاكرة .
لدينا هنا قائمة برنامج التجميع وتذكر أنها على قسمين الأول لتخزين الشاشة :

```
1  F000  21 00 00    LD HL,0000
2  F003  11 00 E0    LD DE,E000
3  F006  01 C0 03    LD BC,03C0
4  F009  CD 59 00    CALL 0059
5  F00C  C9          RET
```

والثاني لاستدعائها وإظهارها على الشاشة مرة ثانية :


```
1  F010  21 00 E0    LD HL,E000
2  F013  11 00 00    LD DE,0000
3  F016  01 C0 03    LD BC,03C0
4  F019  CD 5C 00    CALL 005C
5  F01C  C9          RET
```

البرامج الفرعية في ال ROM التي تقوم بمراقبة النسخ تكون في العنوان 0059 hex و 005c hex (ستة عشر) وكما في المثال السابق فإن السجلات تحتاج إلى أن (تشحن) بالمعطيات قبل أن نقوم باستدعائها . ومهما يكن فالتخزين أو إعادة الاستدعاء للمعلومات التي على الشاشة تحتاج إلى تزويد السجل HL بعنوان المنشأ . فعندما نقوم بتخزين الشاشة (موديل) صفر (Screen 0 فإننا نعلم أن عنوان المنشأ سوف يكون العنوان 0000 .
نلاحظ في السطر 1 أن السجل HL قد (شحن) بالقيمة 0000 . والسجل DE يمثل العنوان المقصود وقد (شحن) بعنوان البداية الذي نريد أن يبدأ التخزين منه في الذاكرة RAM ، وفي السطر الثاني قد (شحن) السجل DE بالقيمة E 000 hex (ستة عشر) . أما السجل BC فهو دائماً يحتوي على عدد البايتات (Bytes) المراد ترحيلها ، وقد (شحن) بالقيمة 03C0 hex (960 عشري) في السطر الثالث .
إذا رغب أحد بنسخ النصف العلوي من الشاشة فقط ولنقل من السطر (0) إلى السطر (11) فالسجل (BC) في هذه الحالة يجب أن يشحن بالقيمة 480 (01 E0 hex) (ستة عشر) السطر الرابع يتم فيه استدعاء البرنامج الفرعي من ال ROM والذي يقوم بنسخ الشاشة وفي السطر الخامس ينتهي البرنامج بتعليمة RET والتي تعني الرجوع إلى برنامج البيزيك .
القسم الثاني ـ استدعاء الشاشة المخزنة ويعمل بطريقة مشابهة مع تعديل السجل HL فقط ، والذي يحوي عنوان المنشج ، وقد (شحن) في الشاشة ذات (الموديل) صفر (Screen 0) بالقيمة 0000 ، في النهاية استدعاء برنامج إعادة الشاشة المخزنة والذي عنوانه 005C . وأما قيمة السجل BC الذي يحوي عدد البايتات (Bytes) المراد نسخها فإن القيمة تبقى 03C0 كما هي .
كما تعلم الآن أصبح لدينا التسهيلات في تخزين واسترجاع إظهارات الشاشة

لنقوم بتعديل البرنامج السابق حتى يزودنا بإمكانات تخزين أربع شاشات. نحتاج فقط إلى تعديل بعض المعطيات في برنامج البيزيك الذي يكتب هذا البرنامج الفرعي في الذاكرة.

عدل هذه الأسطر في البرنامج حتى تصبح على الشكل التالي:

برنامج:

```
20 DEF USR2=&HF020:DEF USR3=&HF030
30 FOR X = &HF020 TO &HF02C
60 FOR X = &HF030 TO &HF03C
```

وفي السطر 50 بدل الرقم السادس والذي هو 224 بالرقم 228 والسطر 80 أيضاً يدل الرقم الثالث والذي هو 224 بالرقم 228

ملاحظة:

يجب أن نكون حريصين عندما نقوم بتعديل سطر موجود على الشاشة. ففي السطر 50 والسطر 80 عندما نقوم بالتعديل يجب ألاّ نضغط المفتاح «RETURN» حتى نأخذ النقطة المضيئة إلى نهاية السطر وبعدها نضغط المفتاح «RETURN» وبغير هذه الطريقة يمكن أن نتسبب بخطأ في البرنامج. استدع قائمة البرنامج قبل أن تنفذه.

بعد تنفيذ البرنامج المعدل أصبح لدينا التسهيلات لتخزين شاشات أخرى في الذاكرة، فقط من هذه المرة يجب علينا كتابة البرنامج الفرعي من العنوان F020، وبرنامج الاستدعاء من العنوان F030 hex (ستة عشر)، والتخزين لهذه الشاشة الثانية يبدأ من العنوان E400 hex. وان تخزين الشاشة الثانية يمكن أن يتم بإدخال: A = USR2 (0).

وإعادة استدعاء الشاشة يتم بإدخال: A = USR3 (0)

وإذا ما احتاج أحد إلى تخزين ثلاث شاشات يجب أن يقوم بالتعديل التالي على البرنامج:

```
20 DEF USR4=&HF040:DEF USR5=&HF050
30 FOR X = &HF040 TO &HF04C
60 FOR X = &HF050 TO &HF05C
```


وفي السطر 50 بدل الرقم السادس والذي هو 228 بالرقم 232 والسطر 80 بدل الرقم الثالث والذي هو 228 بالرقم 232 تحقق الآن من صحة التعديل ثم نفذ البرنامج.

وهذا التعديل يكون من أجل تخزين وإستدعاء أربعة شاشات:

برنامج:

```
20 DEF USR6=&HF060:DEF USR7=&HF070
30 FOR X = &HF060 TO &HF06C
60 FOR X = &HF070 TO &HF07C
```

وفي السطر 50 بدل الرقم السادس والذي هو 232 بالرقم 236 والسطر 80 بدل الرقم الثالث والذي هو 232 بالرقم 236 مرة أخرى تأكد من صحة التعديل ثم نفذ البرنامج.

ـ إن البرامج الفرعية الأربعة لتخزين وإستدعاء الشاشات تم معالجتها بـ:

الاستدعاء	التخزين
A=USR1(0) recalls	A=USR0(0) stores-
A=USR3(0) " "	A=USR2(0) " "
A=USR5(0) " "	A=USR4(0) " "
A=USR7(0) " "	A=USR6(0) " "

الفصل الثاني
تعليمات المعالج - Z 80
Z 80 Instructions

في هذا الفصل سوف نلقي نظرة عامة على طرق ترجمة المعالج Z 80 لأرقام لغة الآلـة، وعلى سجـلات المعـالـج Z 80 والطـرق العـامـة في استخدامها، وبعد ذلك الأنـواع المختلفـة لأوامـر برنـامج التجميع «ASSEMBLER» وسوف نجد قائمة كاملة لهذه الأوامـر المـرمـزة في المـلاحق الموجودة في نهاية هذا الكتاب مفهرسة أبجدياً ورقمياً حسب الحرف الأول لرمز التعليمات.

إن هنـاك العـديد من الكتب المتاحة والتي توضح تعليمات المعالج Z 80 بتعمق أكثـر وعلى الأغلب تكـون موسـوعـة كبـيرة ولكنها غالباً ما تكون كمرجع وليس فيها أسئلة توضيحية (للميكروكومبيوتر) الذي يعمل على نظام الـ MSX بالتحديد.

وعلى كل حال إذا رغب أحد لمعلومات تفصيلية أكثر من تعليمات المعالج Z 80 فعليه شراء الكتاب الذي يختص بهذا الموضوع.

إن لغـة البيـزك BASIC تتألف من أكثـر من مئتي أمـر وبـاختـلافات طفيفة بين تعليـاتها مثـل: «IF-THEN GOSUB» و «IF-THEN PRINT» أمـا لغـة الآلة للمعالج Z 80 فتتألف من حوالي 700 سبعمائة أمر.

ـ لا تخف من هذه العــدد ـ لأن معظم هذه التعليـمات بسيطـة ومتشـابهـة مع اختلاف طفيف بينها.

الفرق الجوهري كما لمسته بنفسك بين لغة البيزيك ولغة الآلة.

إن تعليمـة بيـزيـك واحـدة تقـوم باستدعاء عدة تعليمات رئيسية للغة الآلة في داخل المترجم (Interpreter).

أما إذا ما كتب بلغة الآلة فإنه ينبغي عليك توليد هذه التعليمات بنفسك، وباستطاعتك استدعاء برامج مساعدة ومفيدة موجودة في الذاكرة ROM . كما تفعله بعض البرامج التوضيحية في هذا الكتاب.

من الممكن كتابة برامج من غير أن يكون لدينا المعرفة التامة بالتعليمات والأوامر، وفي الحقيقة كثير من الأشخاص يقومون بهذا العمل بشكل ناجح تماماً حيث يضيفون إلى معرفتهم فوائد هذه الخبرة.

إن هذا الكلام ينطبق إلى حد ما عندما نقوم بالبرمجة بلغة البيزيك (BASIC) . وعلى سبيل المثال ـ كيف تستطيع أن تعد من 1 إلى 1000 بلغة البيزيك؟ لاحظ هذه الطريقة :

برنامج :

```
10 FOR I=1 TO 1000
20 NEXT
30 PRINT "ALL DONE"
```

هذا جيد، ولكن بفرض أنك لا تعرف تعليمة الحلقة FOR-NEXT فمن المحتمل أنك تكتب البرنامج على النحو التالي :

برنامج :

```
10 A=0
20 A=A+1
30 IF A<1000 THEN 20
40 PRINT "ALL DONE"
```

ولكن إفرض أيضاً أنك لا تعرف عن كيفية بناء تعليمة IF-THEN فيجب عليك أن تتغلب على هذه المشكلة وتكتب البرنامج على النحو التالي :

برنامج :

```
10 A=0
20 A=A+1
30 B=-1*(A<1000)-2*(A=1000)
40 ON B GOTO 20,50
50 PRINT "ALL DONE"
```


كما ترى الآن لقد أصبح البرنامج أطول ويحتاج إلى وقت أكبر في التنفيذ إذا لم نستخدم التعليمات المناسبة.

إن المعرفة بجميع التعليمات تساعدك على كتابة برنامج أقصر وأسرع في التنفيذ، وعموماً فإن برامج لغة الآلة سريعة بما فيه الكفاية حتى ولو كتبت بطريقة طويلة. ولكن عند تنفيذ عدد كبير من الأعمال المتكررة كما في برنامج لعبة الشطرنج فإن أجزاء الثانية تأخذ بعين الاعتبار عند تنفيذ البرنامج.

يمكنك القول أن البرامج التي في هذا الكتاب قد كتبت لعرض المفاهيم والمبادىء الأولية فليس من الضروري أن يراعى فيها السرعة أو أن تكون قصيرة ومختصرة في اعطاء النتائج.

ـ ماذا تعني جميع هذه الأرقام؟

إن لغة الآلة تتألف من مجموعة أرقام. فالرقم يعني شيء من اثنين لوحدة المعالجة المركزية للمعالج Z 80 في جهاز (الكومبيوتر) الذي لديك، فإما يعني أمر أو جزء من أمر للقيام بشيء ما. أو يمكن أن يعني جزءاً من معلومات يمكن التعامل معها أو استخدامها بطرق أخرى. ولحسن الحظ أن المعالج Z 80 يعلم تماماً ما يمثله كل رقم بالتحديد (في البرامج المكتوبة بشكل صحيح) ويقوم بالعمل بناء على ذلك.

لنأخذ الأمر الذي يقوم (بشحن) السجل (A) بالرقم «7» مثلاً أي تخزين السجل A بالرقم 7، «سوف نناقش موضوع السجلات بشكل مفصل لاحقاً». ففي لغة التجميع يمكن أن تكتب هذه العملية بشكل رمزي كما يلي: LD A,7 . أما في لغة الآلة فإن هذه العملية تمثل برقمين (ستة عشر) «3E 07» .

فعندما يقوم المعالج Z 80 بقراءة الرقم الأول فإنه يقول «3E تعني أنه يجب عليّ أن أخزن الرقم التالي مباشرة في السجل A» .

فيأخذ الرقم 7 ويضعهُ مباشرة في السجل A وبعد ذلك يقوم بالنظر إلى الرقم الذي يلي الرقم 7 ليقوم بتنفيذ التعليمة اللاحقة، وهكذا حتى ينتهي من غير خطأ أو تشويش. فمثلاً الرقمين (الستة عشر) «3E 3E» ماذا يعنيان، 3E تعني (إشحن)

السجل A بالقيمة 3E والتي تساوي (62) عشري وبعد ذلك يقوم المعالج بالنظر إلى الرقم الذي يلي 3E ليقوم بتنفيذ التعليمة اللاحقة.

نلاحظ أن كل بايت (BYTE) مفردة من المعلومات تحتوي على قيمة تتراوح بين (0 - FF) (ستة عشر) أو (0-255) عشري. فدعنا الآن نلقي نظرة على ذلك بمزيد من التفصيل.

تتألف البايت BYTE من 8 بتات BITS ، كل بت BIT تمثل بعدد ثنائي 0 أو 1.

فالعدد الثنائي 11001001 يمكن أن يمثل هكذا:

رقم البت. :	Bit No: 7 6 5 4 3 2 1 0
القيمة الثنائية. :	Binary Value: 1 1 0 0 1 0 0 1

في أي مكان ظهر به العدد 1 في التمثيل الثنائي تكون قيمته مساوية لـ(رقم البت) 2 اثنتين مرفوعة لرقم البت (BIT) المطابق. وبجمع النتائج بعضها مع بعض نحصل على القيمة العشرية للعدد الثنائي.

إليك الآن طريقة حساب العدد العشري من العدد الثنائي السابق:

```
2 to the power 7  = 128
2 to the power 6  =  64
2 to the power 3  =   8
2 to the power 0  =   1 (any no. to the power 0 = 1)
                    ---
                    201
```

العدد: ـ

2 مرفوعة للقوة 7 = 128

2 مرفوعة للقوة 6 = 64

2 مرفوعة للقوة 3 = 8

2 مرفوعة للقوة 0 = 1 (أي عدد مرفوع للقوة 0 = 1).

201

فالعدد الثنائي 11001001 يساوي إلى 201 في النظام العشري.

وللتحويل من العدد الثنائي إلى العدد (الستة عشر) تقسم ثمانية الأرقام إلى مجموعتين كل مجموعة تتألف من أربعة أقسام. (أربع بتات (BITS) على الشكل التالي:

	الطرف اليساري	الطرف اليميني
رقم البت :	3210	3210
القيمة الثنائية :	1100	1001
	2 ∧ 3 = 8	2 ∧ 3 = 8
	2 ∧ 2 = 4	2 ∧ 0 = 1
	12	9

تذكر أن العدد العشري 12 يساوي C = في العدد (الستة عشر) والعدد (الستة عشر) للعدد الثنائي 11001001 يكوني C9 (C9 = 11001001).

كيف يقوم المعالج Z80 بمعاملة الأرقام ذات 2 بايت (BYTES)

تعطى العديد من الأوامر للمعالج Z 80 لتنفيذ عملية ليست مؤلفة من بايت واحدة مثل ـ «LD A,7» بل قد تكون مؤلفة من بايتين (2 BYTES) وعلى سبيل المثال لنأخذ هذا الأمر من لغة التجميع -«LD HL,49 AFH»-

(ملاحظة: الحرف H في نهاية التعليمة يخبر البرنامج المجمع أن العدد يكون (ستة عشر) hex).

إن الأرقام التي تتألف من بايتين (2 BYTES) تزيد بالقيم العشرية لتستطيع أن تمثل المجال من 0 إلى 65535ـ (hex 0-FFFF) والتي هي بشكل جوهري للعنونة أو لتحديد مواقع الذاكرة التي في جهاز (الكومبيوتر) الذي لديك.

في الأمر LDHL,49 AFH نريد أن نضع القيمة التي ففي البات العليا (49 hex) في السجل H ، والقيمة التي في (البايت) الدنيا (AF hex) في السجل L .

إن رمز هذه العملية في لغة الآلة والتي تعني (شحن) السجل H والسجل L

بالطريقة المباشرة تُمثل بالرقم (21 hex) فعندما يرى المعالج Z 80 الرقم (21 hex) كتعليمة فإنه يأخذ مباشرة الرقم التالي ويضعه في السجل L ثم يأخذ الرقم الآخر ويضعه في السجل H . وهكذا فإن لغة الآلة للأمر LD HL, 49 AFH تكون مشابهة لما يلي : -21 AF 49 hex-

لا شك أنك لاحظت أنه في لغة الآلة ترتيب المعلومات (لبايتين) يكون شكلاً معكوساً (سوف تعرف سبب ذلك الآن) . . .

عندما تستخدم برنامج التجميع فليس عليك أن تقلق لهذه النقطة لأنه هو الذي يقوم بعكسها لك، ولكن عندما تقوم بإدخال لغة الآلة بنفسك كما شاهدنا في الفصل الأول فإن نسيان الترتيب (لبايتات) المعلومات سوف يوقعك بالخطر.

لا حاجة للقول إنه عندما تقوم (بشحن) أي سجل مزدوج بالمعلومات فإن (البايت) الدنيا دائماً تظهر في قائمة لغة الآلة قبل (البيات) العليا.

في لغة التجميع تذكر أن تكتب الرقم بشكله العادي ودع ذلك لبرنامج التجميع ليقوم بوضع الأشياء بالطريقة الصحيحة (سوف نناقش السجلات الزوجية فيما بعد).

نظرة في داخل شريحة المعالج Z 80 — Inside the Z 80 chip

إن العناصر التي تؤلف مجتمعة فيما بينها المعالج Z 80 هي وحدة الحساب والمنطق والتي تقوم بجميع العمليات الرياضية والحسابية البسيطة، وحدة المراقبة والتي تقوم بالتأكد من مرور المعلومات في مفكك الترميز وأنها قد قامت بعملها على أكمل وجه. وهناك أيضاً السجلات الفردية ذات (بايت) واحدة (1 BYTE) أو (8 BIT)، والسجلات الزوجية ذات (بايتين) (2 BYTES) أو (16 BIT) وعلى فكرة فالسجل الفردي ذو (بايت) واحد يمكن أن يستخدم كسجل زوجي ذي (بايتين) أيضاً.

عداد البرنامج : The Program Counter

دعنا ننظر أولاً إلى عداد البرنامج (PC) والذي هو سجل ذوجي من (بايتين) (2 BYTES) .

يحوي هذا السجل عنوان الأمر اللاحق (الذي سوف ينفذ) ويعدل بشكل (أوتوماتيكي) كل مرة بعد تنفيذ أمر جديد. وعلى كل حال فالعنوان المخزن يمكن أن يتغير مثلاً عندما نقوم بإستدعاء أمر ما تماماً مثل تعليمة GOSUB في لغة البيزيك.

في هذه الحالة العنوان الذي في عداد البرنامج (PC) يوضع جانباً في المكدس (STACK) والعنوان الذي استدعي يوضع في عداد البرنامج (PC) . وعند إنتهاء البرنامج الفرعي المستدعى بعد التقائه بالتعليمة (RET) ـ تعليمة الرجوع التي تأخذ رقم مؤلف من (بايتين) من أعلى المكدس (STACK) وتضعه في عداد البرنامج (PC) . عندئذ يتابع التنفيذ من هذا العنوان. وإذا استخدمت المكدس (STACK) فإنه من المهم أن تتذكر أن عنوان الأمر اللاحق بعد تعليمة الرجوع RET يأخذ من أعلى المكدس.

ولعل العديد من البرامج تضل عن التنفيذ لتركها رقماً سهواً في المكدس (STACK) ومن جانب آخر علينا أن نعرف أن عنوان التعليمة اللاحقة الذي في المكدس يمكن أن يكون مفيداً عند نقل المعطيات إلى البرنامج الفرعي.

إن عدداً من الأوامر الأخرى كذلك تؤثر على السجل (PC) عداد البرنامج، مثل تعليمة القفز مثلاً (Jump) ونرمز لها ب «JP أو JR) ولكن في معظم الأوامر طول الأمر (يمكن أن يشمل على أي معلومات لعناصر المعطيات) يضاف إلى عداد البرنامج (PC) بوساطة شرائح نظام المراقبة لذلك فهي تعرف تماماً إلى أين تنظر لمعرفة التعليمة اللاحقة (التي تلي).

مؤشر المكدس The Stack Pointer

لنأخذ سجلاً زوجياً آخر ذا (بايتين) (2 BYTES) مثل مؤشر المكدس (SP) فهو يحافظ على مسار أعلى المكدس بعد إجراء عدة أوامر، كاستخدام المعالج Z 80 للمكدس.

وإن منطقة المكدس تكون داخل ذاكرة الـ RAM لجهازك والعنونة تجهز بوساطة برامج فرعية في الـ ROM بمجرد تشغيل (الكومبيوتر).

وباستطاعتك تجهيز المكدس بعناوين خاصة بك ولكن عليك أن تتذكر بأن

المكدس يعمل بشكل عكسي في الذاكرة ويستخدم نظام (الداخل آخراً ـ خارج أولاً -LIFO- نستطيع تشبيه هذه العملية بكدسة من الصحون موضوعة فوق بعضها البعض باستطاعتك أن تأخذ صحناً من أعلى الكدسة وأن تضع آخر في أعلى الكدسة أيضاً ولكن ليس بإمكانك أن تأخذ صحناً من المنتصف أو في الأسفل فإن ذلك صعب وسوف يسبب تحطيم هذه الصحون.

وهناك نقطة أخرى يجب أن تعرفها على المكدس (STACK) وهي أنه دائماً يستقبل أو يعطي معطيات مؤلفة من (بايتين) (2 BYTES) فإذا وضعنا 11 A0H و 22 B0H و 33C0H في المكدس وبهذا الترتيب فإن مؤشر المكدس يشحن بالعنوان F090 وتكون العملية كالتالي:

عملية:

Address	Contents
F08B	C0
F08C	33
F08D	B0
F08E	22
F08F	A0
F090	11
العنوان	المحتوى

إن مؤشر المكدس في المعالج Z 80 سوف يشير إلى آخر (بايت) (BYTE) (البايت الدنيا) للمعلومة 33 C0H ، وبالنسبة لجزء آخر من المعلومات مؤلف من (بايتين) (2 BYTES) ولنقل مثلاً 4567 H وضعت في المكدس، فإن مؤشر المكدس يُنقص بمقدار واحد وأول (بايت) عليا والتي هي 45 hex (ستة عشر) توضع في العنوان المشار عليه الآن من قبل مؤشر المكدس والذي هو (F08A) وعنوان مؤشر المكدس يُنَقَص مرة أخرى وعندئذٍ (البايت) الأدنى في المعلومة والتي هي 67 hex (ستة عشر) توضع في المكدس في العنوان (F0 89) .

عندما تأخذ معلومة (معطيات) من المكدس فإن النظام يعمل بشكل معكوس ففي مثالنا السابق ان ترتيب (البايت) الأدنى 67 hex ترحل أولاً ومؤشر المكدس يزداد ثم (البايت) ذو الترتيب الأعلى ترحل ومؤشر المكدس يزداد مرة ثانية. وفي هذه المرة أيضاً مؤشر المكدس يشير إلى (البايت) ذي الترتيب الأدنى للمعلومة (33 C0 hex).

السجلات ذات ثماني بتات (8-BIT)

The 8- BIT Registers.

هناك مجموعتان في السجلات ذات ثماني بتات (8- BIT) :

A, F, B, C, D, E, H, L.

.A', F', B', C', D', E', H', L'

ملاحظة: (إن السجل F والسجل F قد وضعا بعد السجل A لأنهما عادة يكونان مرتبطين مع السجل A ولهما أوامرهما الخاصة).

لا تستخدم إلا مجموعة واحدة فقط من السجلات في نفس الوقت، فلماذا لدينا إذاً تكون مجموعتين؟ الجواب يمكنك أحياناً التوقف في منتصف عملية والتغيير إلى المجموعة الثانية من السجلات وتنفيذ عملية ما، وبعد الانتهاء من هذه العملية التي دخلت يمكن التحويل ثانية إلى السجلات الأصلية لمتابعة العملية التي توقفت.

وفي الحقيقة يوجد عدة طرق لمرور المعلومات بين المجموعة الأولى من السجلات والمجموعة الثانية.

يوجد لدينا حقاً عدة سجلات تستخدم كسجلات زوجية لتخزين معلومات مؤلفة من (بايتين) مثل السجل B والسجل C (BC) والسجل D والسجل E (DE) وأيضاً السجل H و L (HL) وفي بعض الأوامر يعالج السجل A و F كسجل زوجي.

السجل The A Register.A

يدعى السجل A بالسجل المركم (Accumolater) فهو غالباً ما يكون لكل الأعمال التي تجري فإنه ـ مثل محطة مركزية عظيمة ـ مشغول دوماً.

فعملياً إن أوامر المقارنة والجمع والطرح (لبايت) مفردة والعديد من الأوامر الخاصة المطلوبة في عمليات النقل والشحن تستخدم السجل A .

السجلان B و C

The B and C Registers

إن العديد من الأوامر تستخدم السجل B أو السجلين B و C معاً كعداد

(للبايت) (Byte Counter) ولنأخذ على سبيل المثال هذا الأمر من لغة التجميع والتي دوماً يجب أن تكون متبوعة بعنونة ما (بطاقة عنوان) : -DJNZ- (نريد في هذا الأمر أن نقول للحاسب نَقِصْ محتوى السجل B بمقدار واحد ، وإذا كانت النتيجة بعد التنقيص ليست صفراً اقفز إلى العنوان الموجود في بطاقة العنوان (Label) وعندما تكون صفراً نفذ التعليمة التي تلي هذا الأمر) .

إن هذا الأمر شبيهاً تماماً بأمر الحلقة FOR-NEXT في لغة البيزيك فمن خلال هذه الحلقة يمكن أن تكرر عدداً من العمليات حسب ما تريد وهذا ما يفعله الأمر (DJNZ) ، حيث يقوم بتكرار تعليمة ما أو عدة تعليمات عدداً من المرات مساوية للرقم الموجود في السجل B .

لنأخذ أيضاً أمراً مشابهاً على سبيل المثال : -LDIR- والذي يستخدم السجل B و C معاً كسجل زوجي ، ويسمح هذا الأمر بنقل كتلة صغيرة أو كبيرة من المعلومات من منطقة في (الكومبيوتر) إلى منطقة أخرى بسرعة قصوى .

ان عدد (البايتات) (BYTES) المراد نقلها بهذه الطريقة يكون موجداً في السجل الزوجي BC .

ملاحظة : يمكن أن نستخدم هذين السجلين معاً (BC) أو كل واحدٍ منهم مستقل عن الآخر وذلك حسب الاستخدامات الخاصة التي تحتاجها .

السجلان D و E

The D and E Registers

هذان السجلان يمكن استخدامها كسجل زوجي أو بشكل مستقل كل واحد منهم عن الآخر، ولعل المعالج Z 80 يستخدمها كمسجل زوجي لتعيين عنوان المقصد .

فلنأخذ الأمر السابق (LDIR) لتوضيح هذه النقطة ، إن عنوان المقصد لنقل كتلة المعلومات من منطقة إلى أخرى يجب أن يؤخذ من السجل الزوجي DE لذلك يجب عليك وضع عنوان المقصد في السجل DE قبل البدء بعملية النقل .

السجلان H و L .

The H and L Registers

يستخدمان هذان السجلان كسجل زوجي لعدد من التعليمات في المعالج Z 80 ، على سبيل المثال الأمر -LDIR- إن عنوان البداية للمعلومات المراد نقلها من منطقة إلى أخرى يؤخذ عادة من السجل الزوجي (HL) فلا تنسى إذاً بوضع عنوان البداية في السجل HL . وهناك أيضاً عدد من الأوامر التي تسمح لك باستخدام السجل HL للإشارة إلى منطقة معلومات .

السجل F أو سجل العلم (Flag)

The F «Flag» Register.

يعتبر هذا السجل من السجلات المهمة بالفعل ، فهو لا يشبه بقية السجلات ذات ثماني (بتات) (8-BIT) الفردية ولا يمكنك تخزين معطيات فيه بالطريقة العادية التي نعرفها . والغرض من هذا السجل هو استقبال نتائج العمليات المنطقية والرياضية ووضع (عَلَم) النتيجة في (البت) المناسبة فيه (فهو مثل الحكم في المباريات الرياضية) . ومن المهم جداً أن بعض سجلات العلم يمكن إختبارها لتجهيز القفز المشروط أو عمليات الإستدعاء والعودة .

ملاحظة : عندما تؤثر بعض الأوامر في جزء من سجل العلم أو فيه كله ، فإنه يبقى على وضعه الحالي حتى تؤثر عليه تعليمة لاحقة .

وكن متأكداً من أن الأوامر المتوسطة لا تؤثر على حالة سجل العلم المشار إليها وهذه الميزة تساعد على تخفيض مقدار الحاجة للترميز .

وعلى سبيل المثال إن اثنان من أوامر الشحن لا يؤثران على سجل العلم أبداً فإذا استدعي واحداً من برنامجين فرعيين معتمداً على حالة سجل العلم الخاصة وعندما يحتاج كل من البرنامجين الفرعيين نفس الشحن في بدايتهما فإن الشحن يمكن أن يحدث قبل أن يتم الإختبار الشرطي .

إن كل بت من بتات سجل العلم «Flag» تدل على وظيفة محددة تقوم بها كما هو مبين في التوضيح التالي :

رقم البايت : Bit Number:	7	6	5	4	3	2	1	0
الوظيفة : Function:	S	Z	-	H	-	P/V	N	C
البت التي يمكن إختبارها : Testable:	*	*				*		*

إن النجمـة في السطـر الأخير تدل على البت التي يمكن إجراء الإختبار عليها بطريقة أو بإخرى من خلال الأوامر المتاحة لذلك .

سوف نرى الآن وظيفة كل (بت) من (بتات) سجل العلم «Flag» .

عَلَمْ الإشارة «S» (في سجل العلم FLAG)

The S OR Sign Flag

إن هذا العلم يقـوم بتكـرار قيمة (البت) الأكثر أهمية في هذا (البايت) في نتائج العمليات الرياضية والمنطقية وعملية الإزاحة . وعند نقل (بايت) إلى السجل A أيضاً يقـوم بتكـرار قيمـة هذه (البت) الأكثـر أهمية في هذا (البايت) والتي هي (البت) ذات المـرتبـة السـابعـة (BIT7) ففي العديد من الأمثلة تستخدم هذه (البت) (ذات الأهمية الكبيرة في سجل العلم) لتشير إلى شرط جزئي محدد .

ففي حالـة الـترقيم المتمم على سبيـل المثـال ، يمثل البت ذو الترتيب السابع إشارة الرقم (موجبة أو سالبة) وهذا يعني أن طول الرقم الثنائي فقط سبعة (بتات) لأن (البت) الثامن كما رأيت هو بت الاشارة . وهذه البتات السبعة تمثل المجال من 128- إلى 128 + ,(128 + — 128-) . في هذا المثال يكون البت السابع يساوي الواحد «1» إذا كان الرقم سالباً ويكون صفراً «0» إذا كان الرقم موجباً .

وكـذلـك يكـون للبت ذات الـترتيب السـابـع دور عنـدمـا يربط البرنامج مع وحـدات الإدخـال والإخـراج مثـل الطـابعة مثلاً . ويمكن اختبار علم الإشارة «S» البت ذات الترتيب السابع (BIT 7) كأي بايت أخرى موجودة .

إن عدداً من أوامر لغة التجميع تسمح باختبار علم الإشارة «S» وذلك بإضافة «P» أي موجب (POSITIVE) أو بإضـافـة «M» أي سالب (NEGATIVE) إلى بعض الأوامـر مثـل تعليمـة القفز مثلاً (JP (JUMP ، فتعليمة القفز الشرطية يمكن أن تحول بإضافة «P» أي موجب إلى الأمر ليصبح : (JP P, Lable) فهذا الأمر يقوم بإختبار علم الإشـارة «S» إذا كان موجبـاً فإن (أي مسـاوي للصفر S = 0)عملية القفز تتكرر وإلا

أي عندما يكون سالباً (أي مساوية للواحد S = 1) تتابع المعالج بتنفيذ الأمر الذي يليه .

علم الصفر «Z»

The Z or ZERO FLAG

يستخـدم هذا العلم للإشـارة إلى نتيجة العمليات الرياضية سواء أكانت صفراً أم لا . أو للإشارة إذا تحققت عمليات المقارنة أو لم تتحقق فعندما تكون نتيجة عملية رياضية تساوي الصفر أو تحققت عملية المقارنة فإن محتوى العلم Z يكون واحداً أي (Z = 1) و عند عدم تحقق عملية المقارنة أو كانت نتيجة العملية الرياضية غير مساوية للصفر فإن محتوى العلم «Z» يكون صفراً أي (Z = 0) .

ويمكن اختبـار علم الصفـر «Z» بإضـافة «Z» (أي هل Z تساوي الصفر؟) أو «NZ» (أي هل Z لا تساوي الصفر؟) إلى أوامر لغة التجميع .

وعلى سبيـل المثـال الأمـر: «RET Z» أي ارجـع عندما يكون Z مساوياً للصفر ويستخـدم هذا الأمـر في الـرجـوع الشـرطي من البرنامج الفرعي فإذا كانت العملية السابقة قد تركت واحداً في علم الصفر فسوف يتم الرجوع من البرنامج الفرعي وإلا فإن المعـالجـة سوف تتـابـع بالتعليمة التي تلي هذا الأمر . كما رأينا حول العَلَم «Z» فلا دَاعي للقلق حول قيمـة (البت) «Z» لأن المعـالـج Z 80 يراقب هذه (البت) ويقـوم بالعمل طبقاً لمصلحتك .

علم نصف الباقي «H»

The H or Half- Carry Flag

يستخـدم هذا العلم خلال عمليـة الـترميـز الثنـائي للعمليـات الـريـاضيـة العشرية ، ليشـير إذا كان هنـاك باقي أم لا وذلـك من (البت) ذي الـترتيب الثالث إلى (البت) ذي الـترتيب الـرابـع (BIT 3- BIT 4) ، ولا يستخـدم هذا العلم في أي نوع من أنـواع الإختبار الشرطي .

علم زيادة التماثل «P/V»

The P/V or parity orerflow Flag

إن لهذا العلم ثلاث وظـائف . ففي بعض الأوامـر يوضـع (0) صفر أو (1) واحد

في العلم P/V وذلـك اعتـماداً على (بـايت) النتيجـة إذا كان عدداً زوجيـاً فإن العلم يوضع به «1» واحد (تماثل زوجي = 1 في العلم)، أو يكون العدد فردي فيوضع «0» صفر في العلم P/V (تماثل فردي = 0 في العلم).

الاستخدام الثاني للعلم P/V هو ليشير خلال عملية الترميز الثنائي للعمليات العشـريـة إذا كان بت الإشـارة «S» (البت الثـامن) قد تأثر بزيادة في (البت) السابع (BIT 6) أو لم يتأثر، وهكذا تتغير إشارة النتيجة بشكل مفاجيء.

وأخيراً فإن خلال أوامـر نقـل الكتل مثل تعليمة «LD IR» يستخدم هذا العلم P/V لمراقبة أي عداد قد وصل إلى الصفر.

ويمكن اختبار هذا العلم P/V بإضافة «PO» والتي تعني (هل التماثل فردي؟) أو «PE» والتي تعني (هـل التـماثـل زوجي) إلى الأوامـر المستخـدمـة في تحويـل تنفيذ البرنامج وعلى سبيل المثال الأمر (CALL) أمر الاستدعاء لبرنامج فرعي يمكن تحويله إلى استـدعـاء شرطي إذا ما أشـار علم التـماثـل إلى رقم فردي وذلك بكتابة التعليمة بالشكل «CALL PO, Label» بدلاً من الأمر الذي ليس مشروط «CALL Label»

عَلَم الطرح «N»

The «N» or Subtract Flag

يستخـدم هذا العَلَم «N» في المعـالـج Z 80 خلال عمليـات الـترميـز الثنـائي للحسابات العشرية الخاصة به ولا يمكن اختبار هذا العلم.

علم الباقي «C»

The «C» Or carry Flag

يقـوم هذا العلم «C» بوظيفتـين، **الأولى**: الإشارة إلى نتيجة عملية الطرح أو الجمـع سواء أكـانت فيهـا استعـارة أم لا، فإذا تكررت عملية الإعارة فإن العلم «C» يوضع به واحد «1» وإلا فسوف يوضع صفر «0» فيه. وبما أن أوامر المقارنة (مثل الأمر (CP B) والذي يوم بمقـارنـة محتـويـات السجل B مع محتويات السجل A) تتم بطرح سجـل معـين من السجـل A (والتخلص من النتيجـة)، وإن العلم «C» يستطيـع الإشـارة إذا كان سجلاً معيناً يحوي قيمة أكبر من السجل A (التي تسبب الباقي) أو يحوي قيمة مساوية أو أصغر من القيمة التي في السجل A (والتي لا تسبب باقي).

الاستخدام الثاني للعلم «C» في الكثير من أوامر التدوير والزحزحة، التي تقوم بترحيل المعطيات على طول (البايت) بإتجاه واحد أو بطرق خاصة أخرى.

من أجل هذه الأوامر يستخدم العلم «C» (كبت) تاسع (يعتبر البت تاسعاً).
وعلى سبيـل المثـال: الأمـر RRA من لغـة التجميـع (يقـوم بتـدوير يميني للسجل A *المركم()) يرحل البت الأول اليميني ذا الترتيب صفر للسجل A إلى علم الباقي «C»، ويـرحـل أيضـاً ما كان في العلم «C» إلى البت ذات الـترتيب السـابـع (BIT 7) في السجـل A ويـرحـل ما كان في البت ذات الـترتيب السـابـع إلى البت ذي الـترتيب السادس وهكذا إلى آخر بت في السجل A.

إن هذا الأمـر الخـاص يدور المعلومـات بشكـل فعـال بتحـويل البتات واحدة واحدة وبمساعدة العلم «C» في هذه المعالجة.

أما في الأوامر المنطقية AND, OR, XOR فإن العلم «C» يبقى دائماً يساوي الصفر «0» لعدم وجود باقي في هذه الأوامر ويبقى السجل A مع ANDA, OR A على حاله ولا يتأثر. بينما في حالة XOR A فإن العلم «C» لا ينظف لوحده فقط بل السجل A أيضاً.
ويمكن اختبـار هذا العلم «C» في الأوامـر الشـرطيـة بإضـافـة «C» أي يوجد باق أو «NC» أي لا يوجد باق إلى هذه الأوامر، وكذلك أمر الاستدعاء يمكن تحويله عندما يكـون هنـاك باق وذلـك بكتـابـة الأمر على الشكل التالي «CALL C, Label» بدلاً من «CALL label».

كيف تستطيع الأوامر أن تؤثر في الأعلام (Flags)

ان الجـدول التـالي يرينـا الأعـلام (Flags) تتأثر بعدة أنواع من الأوامر، وهناك بعض الأوامـر التي ليست بمـدونـة في الجـدول مثل «PUSH» ومعظم أوامر (الشحن) «LD» لا تؤثر في الأعلام (Flags) مطلقاً.
ويـرجى المـلاحظة أن هذا الجدول يحتوي فقط على الأعلام التي يمكن إجراء الاختبار عليها وأن الأمر OR بأشكاله ORC, ORB, ORA . . . جميعاً لها التأثير نفسه على الأعلام ولذلك لم تدون في هذا الجدول.

جدول :

COMMAND	C	Z	P/V	S
ADD A,ADC,SUB,SBC, CP,NEG	?	?	?V	?
AND,OR,XOR	0	?	?P	?
INC,DEC	-	?	?V	?
ADD RR,CCF	?	-	-	-
RLA,RLCA,RRA,RRCA	?	-	-	-
RL,RLC,RR,RRC, SLA,SRA,SRL,DAA	?	?	?P	?
SCF	1	-	-	-
IN	-	?	?P	?
INI,IND,OUTI,OUTD	-	?		
INIR,INDR,OTIR,OTDR	-	1		
LDI,LDD	-		?	
LDIR,LDDR	-		0	
CPI,CPIR,CPD,CPDR	-	?	?	?
LD A,I; LD A,R;	-	?	IFF	?
BIT	-	?		

(FLAGS: C, Z, P/V, S)

رموز الجدول السابق :

? = تعتمد على نتيجة العملية .

P ? = تعتمد على التماثل للنتيجة .

V ? = تعتمد على الزيادة في النتيجة .

0 = محتوى العلم (Flag) يساوي للصفر .

1 = محتوى العلم (Flag) يساوي للواحد

- = العلم (Flag) لا يتأثر: «يبقى العلم على حالته السابقة»

IFF = محتويات مذبذب الإعاقة (FLIP- FLOP)

والفراغ يدل على أن العلم يحوي معلومات غير متعلقة بالأمر .

نلخص فيما يلي الإختبارات الشرطية المتاحة للأوامر JMP Relative, Call, Jump (القفز النسبي)، Return, :

Z = إفعل إذا كانت النتيجة صفراً .

ZN = إفعل إذا كانت النتيجة ليست صفراً

C = إفعل إذا كان هناك أي باقٍ

NC = إفعل إذا لم يكن هناك أي باقٍ

PO = إفعل إذا كان التماثل فردياً .

PE = إفعل إذا كان التماثل زوجياً .

P = إفعل إذا كانت اشارة العلم موجبة (S = 0)

M = إفعل إذا كانت إشارة العلم سالبة (S = 1)

سجلات الفهرس IX و IY

The Index Registers IX and IY

يعتبر هذان السجلان من أكثر السجلات أهمية ذات (الستة عشر بت) (16-BIT) في المعالج Z 80 . وهما لا يشبهان السجلات من A إلى F التي رأيناها فسجلات الفهرس لا يوجد لها مجموعة ثانية كما للسجلات الأخرى ولكن محتوياتها متاحة لكلتا المجموعتين في السجلات A إلى F وA' إلى F'

وإن أوامر (الشحن) المرتبطة بالسجلين IX و IY يجب أن تشمل على مقدار إزاحة . وهذا مما يجعل بيانات الجداول سهلة التركيب باستخدام السجل (IX) أو السجل (IY) لتحديد أساس العنوان ومقدار الإزاحة لتحديد المكان المطلوب بالضبط داخل الجدول، ولعل المثال التالي يساعد على توضيح مهمة السجل IX و IY :

لنفرض أننا قررنا إنشا جدول من المعلومات والذي يحوي عدداً من أسماء الموظفين، مع العناوين وأرقام الهواتف الخاصة بهم . نقوم بتحديد المعلومات التالية :

20 (بايت) لتخزين الإسم، 60 (بايت) لتخزين العنوان، 12 (بايت) لتخزين أرقام الهواتف .

يتألف جدولنا الآن من سلسلة من الكتل كل كتلة بطول 92 (بايت) (60 + 20 + 12) . ونعلم الآن أن أرقام الهواتف لأي اسم تبدأ من البايت الثمانين (80) من بداية الأسم فإذا وضعنا السجل IX على بداية الاسم في الجدول وكما نعرف أن رقم الهاتف سوف يبدأ من (IX + 80) . هذه الطريقة تحفظ عدد (البايتات) للحصول على العنوان الصحيح تماماً .

وإليك الآن البرنامج مكتوب بلغة التجميع :

البرنامج :

```
        LD B,11
        LD IX,NAME3
        LD DE,BUFFER
GETTEL:LD A,(IX+80)
        LD (DE),A
        INC IX
        INC DE
        DJNZ GETTEL
        Next operation
```

ـ الأمر الأول استخدم السجل B كعداد .

ـ الأمر الثاني (شحن) السجل (IX) بعنوان من بايتين (2-BYTES) الذي نحتاجه للإسم NAME3 .

ـ الأوامر التي تلي تقوم (بشحن) وتجهيز السجل DE ليشير إلى منطقة المخزن الوسيط لبيانات رقم الهاتف الذي نريده وتحضيره للإخراج .

ـ نأتي بعد ذلك إلى بداية الحلقة التكرارية التي تقوم بتجميع (بايتات) البيانات من الجدول. نجمع (بايت) واحداً ومن ثم نزيد قيمة السجل (IX) وكذلك نزيد قيمة السجل DE (للإشارة إلى العنوان الذي يلي) ثم نقوم بجمع (بايت) آخر وهكذا حتى يصبح محتوى العداد B مساوياً للصفر.

لاحظ أن الأمر LD A, (IX+ 80) يعني (شحن) السجل A بمعلومات (البايت) التي عنوانها (IX + 80) وكذلك الأمر LD (DE), A يعني شحن معلومات (البايت) الموجودة في السجل A إلى العنوان الموجود في السجل الزوجي DE .

كذلك السجل (IX) يستخدم طبعاً بالطريقة نفسها التي يستخدم بها السجل IX في أوامر (الشحن) (LD.) ويستخدم أيضاً في الأوامر SET, BIT, RLC, INC, ADD .

فالأمر INC (IX + 80) على سبيل المثال يعني الذهاب إلى العنوان المشار إليه بـ (IX + 80) وإضافة واحد إلى (البايت) الذي هناك مهما كان.

لنسأل الآن سؤالاً منطقياً: ما مقدار الإزاحة التي يمكن أن تكون؟ . .

لنوضح نقطة مهمة قبل الإجابة على هذا السؤال، إن مقدار الإزاحة يعالج كرقم ذي إشارة. وهذا يعني يمكن أن يكون بطول سبعة بتات مع تمثيل إشارة المقدار بالبت الأكثر أهمية (بت الإشارة). والجواب لسؤالك الآن أصبح واضحاً ونستطيع **أن نقول: إن مقدار الإزاحة يمكن أن يكون أي قيمة بين** (127 + – 128-) **مع اعتبار أن** الصفر «0» يعالج كمقدار موجب.

السجلين I و R

The I and R Registers

يعتبر هذان السجلان من السجلات ذوات الثمانية بتات (8-BIT) الموجودة في المعالج Z 80 والتي يمكن معالجتها بوساطة الأوامر.

السجل «I» ويعني سجل صفحة المقاطعة (Interrupt-Page Register)

والسجل «R» ويعني سجل تجديد الذاكرة (memory-Refresh Register) .

يستخدم سجل المقاطعة «I» في أسلوب المقاطعة الخاص لعملية ما والتي يمكن أن تتم في المعالج Z 80 بوساطة أمر معين، ويقوم بتخزين البايت الأعلى من العنوان المستدعى عند مقاطعة المعالجة. أما البايت الأدنى فتولد بوساطة جهاز توليد المقاطعة.

دعنا الآن باختصار نقوم بفحص مفهوم **المقاطعة (Interrupt)** . **عندما تكتب** برنامجاً يقوم تماماً بالعمل الذي وضعته من أجله فيقوم بالتفريع وتنفيذ برامج فرعية ويعود بعد ذلك إلى البرنامج الرئيسي كما هو **مخطط** له بالضبط ومهما يكن فقد تراعى طلبات **وحدات الإدخال والإخراج** حتى عندما يكون برنامجك يعمل بشكل تام وصحيح وعلى سبيل المثال معالج وحدة العرض المرئية (VDP) في نظام الـ MSX الذي لديك يعتبر واحداً من هذه الوحدات.

فإشارة المقاطعة ترسل عن طريق الوحدة إلى المعالج Z 80 وكأنها تقول أرجو **الإنتباه إليَّ**. أما برنامجك الرئيسي في هذه الحالة يتوقف ريثما يتم إحضار طلب **المقاطعة**.

ففي حالة معالج وحدة العرض (VDP) المقاطعة من أجل تجديد عرض الشاشة وبعد ذلك تعود المراقبة إلى البرنامج الرئيسي لمتابعة عمله من حيث توقف.

يستطيع أي مبرمج استدعاء معالجة **الانقطاع بنفسه**، وسوف ترى بالفعل «hook» الخطاف) من العنوان FD 9A والعنوان FD 9F يعالج خمسين مرة كثانية واحدة، فالخطاف «hook» هو عبارة عن خمسة (بايتات) في الذاكرة RAM تجهز للرجوع (Returns) (يحوي الخطاف على القيمة C9 للرجوع) وباستطاعة المستخدم أن يستفيد من هذه (البايتات) الخمسة في عملية الاستدعاء (CALL nnnn) إلى برنامج الانقطاع الخاص به ومن ثم الرجوع إلى البرنامج الرئيسي.

وهناك ثلاثة نماذج من الانقطاعات يمكن استدعاؤها عن طريق الأوامر IM1, IM0 و IM2 . ففي الانقطاع نموذج 0 وهو الذي يكون بمجرد تشغيل (الكومبيوتر)، تزود الوحدات الخارجية بأوامر خاصة ليقوم المعالج Z 80 بتنفيذ ما تريده عند طلب الانقطاع.

أما في الانقطاع نموذج 1 عندما تتكرر طلبات الانقطاع فإن قفزة (أوتوماتيكية) تحدث من قبل المعالج Z 80 إلى الذاكرة ذات العنوان (38 hex) (ستة عشر).

إن موقع أي برنامج يقوم بالتنفيذ يكون بالطبع موقع تخزين مؤقت ولذلك فإنه بعد انتهاء (روتين) المقاطعة فإن الرجوع يمكن أن يكون إلى البرنامج الأصلي. وهذا النموذج من المقاطعة دوماً يستدعى إلى العنوان (38 hex) لأن هذا العنوان في نظام الـ MSX يزود القفز إلى (روتينات) مقاطعة معدات (الكومبيوتر) في العنوان **(0C 3C hex)**

أمـا بالنسبة للنمَوذج الثالث من المقاطعة فإنه يعمل بأسلوب مشابه ما عدا بدايته تكون بالذهاب إلى عنوان واحد من أصل 128 عنوان بدلاً من أن تكون عنواناً واحـداً، **كما يزود جهـاز الاستـدعاء يضمّ بمحتويات السجل I . وتجب الإشارة إلى** أن البت ذي الترتيب صفر (BIT 0) أي الأول من بايت العنوان الذي يأتي من جهاز الاستدعاء دوماً يكون صفراً (0) .

إن العنوان المشار اليه **إضـافـةَ إلى** العنوان الذي يلي يزود بعنوان من (بايتين) (2-BYTE) يكون في متناول (روتين) الانقطاع لتحديد أي المراقبات التي قد مرت.

أما في بعض البرامج فإنه يكون من الضروري التأكد من أن الانقطاع لم يتكرر خلال عملية محددة وان الأمر (DI) أمر عدم السماح بالانقطاع يساعدك على هذه المهمة ولكن أرجو أن لا تنسى أن تعيد عملية السماح بالانقطاع من خلال الأمر (EI) عندما تنتهي في هذا الجزء من برنامجك.

نستطيع القول في النهاية أن السجل (R) سجل التجديد هو لتجديد الذواكر بشكل (أوتوماتيكي). ويمكن استخدام هذا السجل على أنه نوع من أنواع برامج التوقيت ولكن تذكر أن قيمته تبدأ من الصفر (0) إلى 255 عشري ولذلك فإنه لا يعتبر بالسجل المفيد في هذا النوع من البرامج.

أوامر التجميع :
THE ASSEMBLY COMMANDS

هناك عدة طرق لتصنيف العديد من أوامر التجميع سوف نجمعها لك تحت خمسة عناوين لتغطي كافة الأوامر التي:

١ ـ تنقل المعلومات من مكان إلى آخر.
٢ ـ تعالج وتختبر المعلومات بطريقة ما.
٣ ـ تحول مسار برنامج ينفذ بشكل تسلسلي.
٤ ـ تستخدم وحدات الإدخال والإخراج.
٥ ـ تراقب النظام.

قبـل أن نخـوض في هذه الأوامر لعله من المفيد أن نستعرض بنظرة سريعة الطريقة التي ينفذ بها الأمر في المعالج Z 80

إن كل أمر ينفذ بثلاثة أطوار. في الطور الأول، يتم إحضار الأمر من مكانه الصحيح في البنامج وعداد البرنامج يخبر المعالج Z 80 أين يقوم بالبحث و(البايت) الأول للأمر يوضع عندئذ في سجل يحتفظ به المعالج Z 80 لنفسه يدعى سجل الأمر.

في الطور الثاني يكون الأمر عندئذٍ قد فُكِكَ من قبل المعالج Z 80 ويقوم بتجهيز دورة العملية للطور الثالث وهي التي تكون في العادة تنفيذ الأمر.

إن كل طور يعمل بخطوات محددة تماماً يدعى دور ساعيَّةً أو (حالة زمنية). .

والدورات التي تعمل بنفسها تدعى دورة الحاسب وإن أقصر دورة حاسب هي آخر ثلاثة دورات سَاعيَّةً، وكل دورة تعني وحدة مستقلة من الزمن. الدورات التي تحتاج إلى أوامر أكثر لجلبها (أوامر التفكيك والتنفيذ مثلاً) يؤخذ الأطول منها فينفذ.

والفكرة من كل ذلك: أن الاوامر الطويلة تأخذ للتنفيذ ولكن مهما يكن فإن تعقيد الأمر أيضاً يلعب دوراً في ذلك، فإن بعض الأوامر تأخذ وقتاً في التنفيذ أكثر

من بعضها الآخر مع أن لها طول البايت نفسه على سبيل المثال: **الأمر المؤلف من** (بايت) واحد لإنقاص السجل -DEC BC- يأخذ دورة حاسب واحدة وست حالات زمنية (6 T-States) ، بينما الأمر -DEC A- مؤلف من (بايت) واحد أيضاً ويأخذ دورة حاسب واحـدة ولكن مع أربـع حالات زمنية (4-T-States) أي أنه أسرع بحالتين زمنيتين (2T-States) أو (ميكرو) ثانية واحدة إذا كانت الساعة تعمل على 2MHz أو حتى أقل من 3.53 MHz في نظام الـ MSX .

تعتبر هذه المناقشة حول دورة الحاسب والحالة الزمنية كافية تماماً لما سوف ندرس عن لغة الآلة وان الغرض من هذا الكتاب هو مناقشة السرعة الطبيعية أو العادية لكل أمر.

وكما أشرنا سابقاً أن معظم برامج لغة الآلة تنفذ بسرعة كافية تماماً بدون أي نقص يجرى عليها.

اصطلاح الأقواص () :

The «Brackets Convention»

قبل أن تبدأ بالأوامر هناك إصطلاح يجب أن يكون واضحاً **تمامـاً بالنسبة لنا** وهو استخدام الأقواص () داخل الأوامر.

إن أي عنوان يمكن الإشارة إليه بطريقتين. فإذا أردنا العنوان نفسه فإنه يكتب بالطريقة العادية -1234 h- على سبيل المثال. وإذا أردنا الإشارة إلى **محتوى** العنوان فإنه في هذه الحالة يجب إحاطة العنوان بقوسين.

الأمر -LD HL, 1234 H- يعني (اشحن) السجل HL **بالعنوان 1234 h.**

لا بد أن تذكر من المناقشة السابقة أن (البايت) الأدنى يوضع في السجل (34 hex) L والبايت الأعلى يوضع في السجل H (12 hex) .

أما الأمر -LD HL, (1234 H)- فهو يعني اذهب إلى العنوان (1234 hex) ومهما يكن (البايت) الذي تجده فيه ضعه في السجل L ثم إذهب إلى العنوان الذي يلي أي (1235 hex) ومهـما يكن البايت الذي تجده فيه ضعه في السجل H . (إذا نظرت إلى عدة صفحات سابقة ستجد كيف يقوم المعالج Z 80 بتخزين السجلات حسب متطلباته).

فإذا كان العنـوان (1234 hex) يحوي (البايت) **hex** (89) والعنوان (1235 hex) يحوي البايت hex (67) فعندئذٍ السجل HL سوف يحوي القيمة (6789 hex) بعد هذا الأمر.

بالمثل لنأخذ الأمر -LD A, (HL)- وهذا يعني اذهب إلى العنوان المشار اليه في السجل HL وضع البايت الذي تجده هناك في السجل A فإذا كان السجل HL يحوي العنوان (1234 H) فأي بايت كان في هذا العنوان (يشحن) إلى السجل A (في مثالنا السابق كانت 89 hex) .

إذا كان السجل HL يحوي (6789 hex) كما في المثال الثاني سابقاً فعندها مهما يكن البايت الذي في العنوان (6789 H) (يشحن) إلى السجل A .

لاحظ أن الأمـر -LD A, HL- لا يمكن أن يكون موجوداً لأنك سوف تحاول (شحن) (بايتين) من المعلومات في مخزن مؤلف من (بايت) واحد، وحتى نظام الـ MSX لا يستطيع القيام بذلك.

١ ـ أوامر نقل المعلومات :

Data Transfer Commands

في هذا الفصـل سنلقي نظرة على جميع الـطرق المختلفـة التي باستـطاعتك استخدامها لزحزحة (بايت) أو أكثر من مكان في الذاكرة إلى مكان آخر وهذا يشمل زحزحة المعلومات حول السجلات فيما بينها.

ويشمل على انشاء معلومات جديدة وهذا يكون (بشحن) سجل بقيمة معينة وغالباً ما تكون قيمة **لإيجادها في مكان آخر في الذاكرة RAM .**

وإن مالم تضُمّن به هذا الفصـل أوامـر القـراءة والكتابة على وحدات الإدخال أو وحدات الإخراج.

ولعل هناك نقطة سنجعلها واضحة تماماً وهي أن المعلومات تبقى في العنوان أو **في السجـل حتى يكتب فوقهـا . وهكـذا إذا قمنـا (بشحن) السجل A من السجل B LD A,B فإن كلا السجلين A و B يحويـان المعلومـات نفسهـا التي كان يحويهـا السجـل B** أما المعلومات التي في السجل A فقد ضاعت.

مجموعة (شحن) السجلات ذات ثمانية بتات :

The 8- BIT Load Group

إن جميع السجلات المؤلفة من ثمانية بتات تنقل بشكل مباشر عن طريق أمر (الشحن) الذي يأخذ الصيغة التالية :

(LD destanation, Source)

(المصدر، المقصد LD)

وعلى هذا النمط يكون المثال التالي : -LD B, D-

وهذا يعني (إشحن) محتويات السجل D إلى السجل B .

ولعل الجدول التالي يرينا أوامر (شحن) السجلات ذات الثمانية بتات الممكنة : الجدول :

مصدر الشحن
Source of the load

مقصد الشحن Load Dest.	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	r
A	x	x	x	x	x	x	x	x	x	x	x	x	x	x
B	x	x	x	x	x	x	x	x			x	x		x
C	x	x	x	x	x	x	x	x			x	x		x
D	x	x	x	x	x	x	x	x			x	x		x
E	x	x	x	x	x	x	x	x			x	x		x
H	x	x	x	x	x	x	x	x			x	x		x
L	x	x	x	x	x	x	x	x			x	x		x
(HL)	x	x	x	x	x	x	x							x
(BC)	x													
(DE)	x													
(IX+d)	x	x	x	x	x	x	x							
(IY+d)	x	x	x	x	x	x	x							x
(nn)	x													

* السجلات التي من جهة اليسار تمثل مقصد (الشحن) أما السجلات التي في الأعلى تمثل مصدر (الشحن) والعلامة «X» تشير إلى إمكانية استخدام الأمر:

ـ المصدر، المقصد LD- بين هذه السجلات .

وهكذا فإذا نظرنا إلى السطر الأعلى عبر سجلات المصدر فإننا نرى الأوامر التي يمكن استخدامها مثل : LD A,A و LD A,B و LD A,C .

وتجب الملاحظة من خلال الجدول أنه لا يوجد أمر (شحن) بين السجل D والسجل الزوجي BC ((LD D, (BC)- «لا يوجد أمر هكذا» ـ ولكن ليس هناك مشكلة .

في الجدول نلاحظ «nn» وهي تعني رقمًا مؤلفاً من (بايتين) (2- BYTE) والذي يمكن أن يمثل عنواناً ما . ولا بد أنك لاحظت من الجدول أن السجل A فقط هو الذي يمكن (شحنه) بمحتويات عنوان معين -(LD A, (nn)- «السطر الأول من الجدول» .

ونلاحظ في السطر الأخير من الجدول أيضاً فقط السجل A الذي يمكن نقل محتوياته إلى عنوان معين، فدعنا نناقش هذا التشعب في هذه النقطة .

إذا أردت أن (تشحن) عنواناً معيناً بمعلومة (بايت) فإنك يمكن أن تقوم بذلك بطريقتين إما بوضع معلومة في السجل A ثم تنفذ الأمر (LD (nn),A) حيث nn العنوان المطلوب .

أو إذا ألقيت نظرة على السطر الأفقي للسجل (HL) تلاحظ أنه إذا (شحن) السجل HL بالعنوان المطلوب -(LD HL,nn)- «**سنأتي على ذكر** هذا الأمر فيما بعد»، فعندئذٍ المعلومات من أي سجل من السجلات A,B,C,D,E وحتى H و L يمكن أن (تشحن) إلى هذا العنوان المطلوب بإستخدام الأمر ـ «**سجل**», LD (HL) .

إذا قمت بدراسة الجدول السابق فإنك سوف ترى أن التطبيقات نفسها بشكل **معكوس تمكنك من (شحن)** أي سجل بالإضافة إلى السجل H و السجل L من عنوان محدد عن طريق السجل HL «السطر العامودي لـ «HL» » . وهكذا فإنك تستطيع أن تكتب -(LD C, (HL - وهذا يعني اشحن السجل C بمحتويات العنوان المحدد بالسجل HL . وتلاحظ أن الطريقة سهلة عندما تعرف كيفية استخدامها .

دعنا الآن نلقي نظرة بشكل آخر على هذا الجدول فنرى أن العمود الموجود في الجانب اليميني من الجدول والذي يحوي «n» وكما يبدو لك أن «n» تعني معلومة من (بايت) واحد تتراوح قيمتها بين 0 إلى 255 عشري أو 0 إلى FF (ستة عشر) .

لاحظ الآن كيف يمكننا (شحن) بايت معين من المعلومات إلى عنوان مشار إليه في السجل HL بالأمر -(LD (HL),n)- .

هناك أربعة من الأوامر التي لم يشر إليها الجدول كنا قد ناقشناها سابقاً والتي لا نحتاج إليها في الحاضر وهي :

(اشحن السجل A من سجل المقاطعة) LD A,I
(اشحن السجل A من سجل التجديد) LD A,R
(اشحن سجل المقاطعة من السجل A) LD I, A
(اشحن سجل التجديد من السجل A) LD R, A

مجموعة (شحن) السجلات ذات الستة عشر (بت)
The 16-BIT Load Group.

الصيغة الأساسية في (شحن) المعلومات المؤلفة من ستة عشر (بت) بايتين (2-BYTE هي نفسها التي تستخدم في (شحن) ثمانية البتات (8-BIT) :
المصدر، المقصد LD
LD destination, Source

ومهما يكن لا بد من وجود بعض الاستثناءات الهامة التي سنأتي على ذكرها.
بعد أن تحدثنا حول (شحن) المعلومات المؤلفة من (بايتين) (2-BYTE)، فإما المصدر وإمَّا المقصد يجب أن يكون بالطبع سجلاً زوجياً.
ولعل الجدول التالي يرينا الأوامر التي يمكن استخدامها في الصيغة «المصدر، المقصد LD» :

Source of the load مصدر الشحن

Load Dest. مقصد الشحن	BC	DE	HL	SP	IX	IY	nn	(nn)
BC							x	x
DE							x	x
HL							x	x
SP			x		x	x	x	x
IX							x	x
IY							x	x
(nn)	x	x	x	x	x	x		

إنه لا يبدو بالجدول المعقد أليس كذلك؟ إن المراد من هذا الجدول تبيان أنه لا يمكنك على سبيل المثال (شحن) سجل زوجي مباشرة مثل السجل BC من محتويات السجل DE . وهذا مايظهره الجدول لأنه لايوجد أمر -LD BC, DE- ولكن كما سوف نرى هذه ليست مشكلة .

نرى في هذا الجدول «nn» والتي تمثل بايتين من المعلومات (2-BYTE) والتي يمكن أن تكون عنواناً أورقماً بسيطاً لبعض العمليات الرياضية، بينما «(nn)» تمثل محتويات العنوان «nn» .

والمهم جداً أن نلاحظ في هذا الجدول غياب السجل A من بين هذه السجلات الزوجية، وفي الحقيقة فإن سجل مؤشر الكدسة /SP/ يمكن شحنه من محتويات السجل الزوجي HL ، أو من سجلات البايتين (2-BYTE) IX أو IY أو بالعنوان المباشر «nn» أو من محتويات عنوان محدد «(nn)» . وهكذا كما ترى لدينا عدة طرق لتجهيز مؤشر الكدسة أو حتى لتبديله أثناء عمل البرنامج . ولكن العكس غير صحيح، فأوامر (الشحن) -LD- تهتم فقط (بشحن) عنوان الـ SP (مؤشر المكدسة) إلى «(nn)» لحفظ قيمته .

والآن نتحدث عن الطرق الأخرى التي لدينا لنقل معلومات من بايتين (2-BYTE) وماذا أيضاً عن سجلنا القديم A ؟ إن باستطاعة الجدول أن يرينا عموداً آخر وسطراً آخر يحوي (SP) فعلى سبيل المثال الأمر -LD (SP), BC- والأمر LD BC, (SP) صحيحان ويمكن تنفيذهما ولكننا لا نستطيع أن ندرجهم مع هذا النوع من الأمر.

دعنا الآن نرى ماذا يعني الأمر LD (SP), BC . أولاً (SP) يعني محتويات عنوان قد سمي في سجل مؤشر الكدسة ـ هذا في أعلى الكدسة، إذاً الأمر -LD (SP), BC- ماذا يعني؟ إن هذا الأمر يعني ضع محتويات السجل الزوجي BC على الكدسة وكذلك بالمثل فإن الأمر -LD BC, (SP)- يعني (اشحن) السجل الزوجي BC من محتويات أعلى الكدسة . في كلتا الحالتين العنوان الموجود في سجل مؤشر الكدسة يعدل بعد نقل كل (بايت) (راجع مناقشة مؤشر الكدسة الذي مر معنا سابقاً) .

هناك أمر مهمته وضع محتويات سجل زوجي على الكدسة وآخر ليأخذ بايتين من الكدسة، وهما الأمر Pus H والأمر PoP :
AF, BC, DE, HL, IX, IY

فإذا أردت تخزين محتويات السجل الزوجي DE على الكدسة فاكتب الأمر PUSH DE وخذ معلومات من أعلى الكدسة وضعها في السجل DE . وإكتب PoP DE . DE-

لا بد أنك لاحظت السجل الزوجي AF حيث باستطاعتك أن تستخدم معه الأمر Pus H والأمر PoP ولهذا فإنك تستطيع بشكل مناسب أن تضع جانباً المعلومات المهمة في كلا السجلين أو أحدها A والسجل F (سجل العلم).

والآن نتحدث عن شحن السجل BC من السجل DE الذي سبق عنه الكلام.

يمكننا القيام بهذه العملية بطريقتين، الأولى: بوضع معلومات DE في المكدس Pus H DE وبعدها عن طريق الأمر PoP BC أي قراءة (بايتين) من أعلى المكدس ووضعهم في السجل BC ، والطريقة الثانية: تستخدم أمر (الشحن) (لبايتين) مفردين LD B, D و LD C,E وكل واحدة من الطريقتين تعمل بشكل جيد.

ان كلتا الطريقتين عبارة عن أمرين طول الواحدة (بايت) واحد وتستخدمان بشكل شامل تماماً. ولكن طريقة الأمرين PUSH و PoP تجعل المعالج Z80 ينظر عبر نفسه وفي منطقة الذاكرة RAM لينفذ الأوامر. بينما طريقة (شحن) سجل تلو الآخر لا تقوم بذلك.

ولكنَّ طريقة (شحن) سجل تلو الآخر تكون أسرع (بحوالي 16 حالة زمنية).

وإذا أردت وضع (بايتين) من المعلومات التي في إحدى سجلي الفهرس IX أو IY في سجل زوجي فلا خيار لك إلا أن تذهب عن طريق الكدسة (STACK). لاحظ إن لم تحدد مقدار الزحزحة للسجلات فإنه سوف ينفذ:

PUSH IX وليس PUSH IX + d

وهناك بعض الأوامر التي تسمع لك بزحزحة (بايتين) من المعلومات من مكان إلى آخر. تدعى هذه الأوامر «المبادلات» (Exchanges) . وهي كما يلي:

EX (SP), HL
EX (SP), IX
EX (SP), IY
EX DE, HL

EX AF, AF'
EXX

إن التبادل يختلف عن (الشحن، لأنه تقوم بالتبديل بين محتويات مكانين معينين. ولهذا فإن الأوامر الثلاثة الأولى تقوم بتبديل المحتويات أعلى الكدسة مع السجلات النسبية مثل: IX, HL وأيضاً IY .

على سبيل المثال، عندما يستدعى أحد (الروتينات) الفرعية (عن طريق الأمر CALL) فإن عنوان الأمر الذي بعد أمر الاستدعاء (CALL) يوضع على الكدسة (STACK) . لأن هذا العنوان سوف يوضع مرة ثانية في عداد البرنامج عندما ينتهي (الروتين) الفرعي ويتم الخروج عن طريق الأمر -RETURN- .

ولكن افرض أننا أردنا ألاّ نضع بعد أمر الإستدعاء (CALL) أمراً تالياً بل وضعنا معلومة أو مجموعة من المعلومات التي نرغب بتمريرها إلى (الروتين) الفرعي.

في هذا (الروتين) الفرعي ينفذ الأمر HL و EX (SP) فتصبح لدينا محتويات السجل HL في أعلى الكدسة (STACK) وما كان في أعلى الكدسة أصبح في السجل HL (العنوان الذي يدل أين توجد المعلومات) ويمكننا الآن إلتقاط هذه المعلومات بالأمر LD A,-(HL)- ونقوم (وهذا مهم جداً) بزيادة السجل HL حتى يتجاوز (بايتات) المعلومات إلى عنوان الأمر التالي وعندئذٍ نقوم مرة أخرى بتنفيذ الأمر EX (SP), HL من جديد.

أصبح لدينا الآن العنوان الصحيح للأمر الذي بعد الأمر (CALL) عندما تخرج من (الروتين) الفرعي (RETURN) جاهزاً في المكان المناسب ليقوم عداد البرنامج بالتقاطه.

بهذه الطريقة نكون قد مررنا المعلومات في (الروتين) الفرعي ليتم معالجتها.

وتعتبر هذه الطريقة الوحيدة لتمرير المعلومات ضمن (الروتين) الفرعي وهي طريقة ناجحة جداً.

الأمر EX DE, HL يعتبر أمثل عندما نقوم بالعمليات الرياضية أو عندما نريد المبادلة بين عنوان المقصد الذي في السجل DE وعنوان المصدر الذي في السجل HL أما الأمر EXX يقوم بتبديل محتويات السجلات الزوجية الثلاث DE-BC و HL مع المجموعة الثانية من السجلات المكملة DÉ - BC' و HL' . ولكن يجب، الملاحظة أن السجل AF له الأمر الخاص به للمبادلة مع سجله المكمل وهو -EX AF, AF'- .

ان المعلومات الموجودة في المجموعة الثانية من السجلات لا تقوم بأي عمل ، تكون مجمدة ، ولهذا فإنك تحتاج إلى طريقة أخرى لتخزين المعلومات بشكل مؤقت بدون تجهيز أي مخازن جديدة أو استخدام الكدسة (STACK) .

وعلى أي حال فإنك ستجد في بعض (الكومبيوترات) أن المجموعة الثانية من السجلات تستخدم بشكل واسع في استعمال (روتين) المقاطعة ، وإلى غير ذلك . ويجب الانتباه إلى أن حذف بعض المعلومات سهواً أو ترك بعض المعلومات الغريبة في المجموعة الثانية من السجلات يؤدي إلى حدوث بعض الأمور المستغربة .

مجموعة نقل الكتلة .

The Block Transfer Group

لقد أتينا الآن على مجموعة الأوامر التي تسمح بنقل عدد من (بايتات) المعلومات من مكان في الذاكرة RAM إلى مكان آخر.

وإليك هذه الأوامر مع وظائف كل منها :

LD I : (اشحن) (DE) من (HL) وزائد السجل DE والسجل HL وناقص السجل BC

LD IR : (اشحن) (DE) من (HL) وزائد السجل DE والسجل HL وناقص السجل BC وكرر العملية حتى يصبح السجل BC = 0

LD D: (اشحن) (DE) من (HL) وناقص السجل DE والسجل HL والسجل BC

LD DR: (اشحن) (DE) من (HL) وناقص السجل DE حتى يصبح السجل BC = 0 .

إن جميع هذه الأوامر تقوم بنقل (بايتات) المعلومات الموجودة في عنوان مشار إليه بالسجل الزوجي HL إلى عنوان مشار إليه في السجل الزوجي DE .

وبعد نقل كل معلومة يتم انقاص القيمة الموجودة في السجل الزوجي BC

والواضح أن هذه السجلات الزوجية الثلاثة يجب أن توضع فيها القيم الأولية قبل قيام أمر نقل الكتلة بالتنفيذ .

في حالة الأمر LDI والأمر فإن السجل HL والسجل DE يتزايدان بعد كل عملية نقل ، بينما في الأمر LDD والأمر LDDR فإن السجل HL والسجل DE يتناقصان بعد كل عملية نقل تحدث . وهذا مما يجعلنا نقول : إن السجل HL والسجل DE يشيران دائماً إلى العنوان الصحيح (لبايت) المعلومات التالية التي يجب نقلها .

ومع الأمر LDIR والأمر LDDR ، ينفذ الأمر الذي يلي بعد كل عملية نقل تتم وهذا يسمح بالقيام بعدة أعمال قبل البدء بعملية النقل مرة أخرى ، ولهذا يجب التذكر بعدم مس أو تخريب القيم الموجودة في السجلين DE و HL أو السجل BC . إن الأمرين LDI و LDD يجعلان قيمة العَلَم P/V صفراً إذا استمرا في إنقاص السجل BC إلى الصفر. والبرنامج التالي يقوم بنقل فقط تلك (البايتات) من المعلومات التي يكون فيها (البت) أكثر أهمية (البت رقم 7 (BIT 7) مساوٍ للواحد «1» .

يفترض هذا البرنامج أن السجل DE والسجل HL قد جهزا بعنواني المقصد والمصدر (البداية) والسجل BC قد جهز بالحد الأعلى لعدد (البايتات) التي سيتم إختبارها ونقل البايت الذي (البت) رقم 7 (BIT 7) فيها مساوياً للواحد .

```
NEXT:LD  A,(HL)     ;Get 'next' byte
     BIT 7,A        ;Test top bit
     JR NZ,MOVE     ;Byte wanted - shift it
     INC   HL       ;Byte unwanted - increment HL
     DEC   BC       ; and decrement the counter BC
TEST:LD  A,B        ;Check if BC is zero
     OR    C        ;by ORing B with C
     JR NZ,NEXT     ;Do it again if BC not zero
     JR    DONE     ;BC is zero - so finish
MOVE:LDI            ;Move the byte
     JP PE,NEXT     ;Do again if BC not zero
DONE:Your next command...
```

احضار (البايت) الذي تلي

اختبار (البت) رقم 7 (BIT 7)

إذا كان (البايت) المطلوب إنقله .

إذا كان البايت غير مطلوب

زيارة السجل HL

إنقاص سجل العداد BC

اختبار هل السجل BC مساوي الصفر ومقارنة B مع C

كرر العملية مرة ثانية إذا كان BC ليس صفراً

BC مساوي للصفر ـ الإنتهاء

ترحيل (البايت)

كرر ثانية إذا كان السجل BC ليس صفراً .

بدلاً من الأمر -JPPE, NEXT- بعد الأمر LDI يمكن القيام بقفزة نسبية إلى الخلف للنقطة TEST والتي تقوم باختبار السجل BC إذا وصل إلى الصفر «0» بعد اجراء التنقيص منه ـ والملاحظ أنه لا يمكن القيام بالقفز النسبي (JR Label) عندما نقوم بالاختبار من أجل التماثل . ولكن المزيد عن هذا الموضوع وعن الأوامر الأخرى مثل (BIT 7,A) و INC و DEC وسوف نناقشها فيما بعد .

ربما خطر لك السؤال التالي : لماذا نحتاج كل من الأمرين LDIR و LDDR ؟ في الحقيقة حتى لا نقوم بالكتابة فوق المعلومات التي نريد زحزحتها ، إفرض على سبيل المثال نريد زحزحة كتلة من المعلومات مؤلفة من (1001 H) من (البايتات) ، من العنوان (8000 H) إلى العنوان (8500 H) .

فإذا استخدمنا الأمر LDIR مع الإشارة إلى أن السجل HL يشير إلى العنوان (8000 H) عنوان البداية والسجل DE يشير إلى عنوان النهاية (8500 H) . فإن (البايت) الأولى سوف تنقل من العنوان 8000 إلى العنوان 8500 H وفي هذه الحالة نكون قد كتبنا فوق كتلة المعلومات المؤلفة من (1001 H) (بايت) التي نريد نقلها .

في مثل هذه الحالة من الأفضل استخدام الأمر LDDR وجعل السجل HL يشير إلى نهاية الكتلة التي نرغب بإزاحتها ولتكن (9000 H) والسجل DE يشير إلى نهاية

المنطقة المراد النقل إليها ولتكن (9500 H) ، ومع تكرار العملية وتناقص السجل DE حتى يصل إلى (9000 H) نكون قد نقلنا المعلومات من تلك المنطقة ولا ضرر إذا كتبنا فوقها .

٢ ـ معالجة المعلومات & أوامر الإختبار.

Data manipulation & test commands

ـ مجموعة الثماني بتات (8- Bit) الحسابية والمنطقية .

إن العمليات الحسابية البسيطة التي يمكن أن تجرى على بايت مفردة تكون بإضافة واحد إليها عن طريق الأمر (INC) أو تنقيص واحد منها بالأمر (DEC)

وهذه العمليات يمكن إجراؤ ها على السجلات التالية والعناوين التي يمكن أن تشير إليها هذه السجلات :

A, B, C, D, E, H, L, (HL), (IX + d), (IY + d)

إن العَلَم Z و P/V والعَلَم S يتأثران طبعاً لنتيجة العملية .

بقية العمليات في هذا الفصل جميعها تجرى على السجل A : ومصادر (بايت) المعلومات الأخرى (حتى ولو كان السجل A أيضاً) يجب أن توصف :

المصادر التالية يمكن أن تستعمل (لبايت) المعلومات الأخرى :

A, B, C, D, H, L, (HL), (IX + d), (IY + d), n

حيث n بالطبع تمثل قيمة محددة .

وإن الأوامر التي يمكن إجراؤ ها هي :

ADD A; ADC A; SUB; SBC; AND; OR; X OR; CP

وسوف نقوم بشرح كل أمر على حدة :

ADD A :

مثال (ADD A,2; ADD A,(HL); ADD A,B) لاحظ أن السجل (A يجب أن يكون معيناً .

كما رأينا هذه الأوامر عبارة عن جمع بسيط (لبايت) معلومات معينة إلى محتويات السجل A . فالأمر (ADD A,(HL يعني اجمع محتويات العنوان المشار إليه بالسجل HL

إلى تلك المحتويات التي في السجل A وضع النتيجة في السجل A . فإذا كانت النتيجة تزيد عن (FF hex) أو (225 عشري) فإن عَلَم الباقي «C» يعطى القيمة «1» (يشير إلى وجود باقي) والسجل A يحوي على النتيجة مطروح منها 256 .

فإذا كانت القيمة (FF hex) في السجل A فبعد الأمر (ADD A,2) تصبح النتيجة في السجل A تساوي الواحد «1» (1 = 256 -257) ويوضع واحد في علم الباقي «C» ليشير إلى أن هناك باقياً .

وكذلك أيضاً العَلَم Z والعلم P/V و S يتأثران طبقاً لنتيجة عملية الجمع .

مثال :

(ADC A,B; ADC A, (HL); ADCA,2)

يعتبر هذا الأمر نفس الأمر ADD تماماً ما عدا أن محتويات سجل الباقي قبل بدأ العملية تجمع إلى السجل A . وعلى هذا النمط إذاً وضع «1» في عَلَم الباقي «C» والسجل A يحوي (21 hex) وبعد الأمر (ADC A,2) فإن النتيجة في السجل تكون تساوي إلى (24 hex) ولأن العملية لا تحتاج إلى باقٍ فإن علم الباقي «C» سوف يعود إلى وضعية الصفر «0» (للإشارة إلى أنه لا يوجد باقٍ) .

: SUB

مثال : ـ (SUB B; SUB, (HL); SUB 2)

نلاحظ أن السجل A لم يوصف في هذا الأمر (مالم تكن تريد تنفيذ (SUB A) أي اطرح محتويات السجل A من A) . هذا الأمر يقوم بطرح معلومات محددة من السجل ويضع النتيجة في السجل A أيضاً وكما في الأمر «ADD» فإن الأعلام (Flags) تتأثر طبقاً لنتيجة العملية .

: SBC

مثال : (SBC B; SBC, (HL); SBC 2)-

الأمر SBC يشابه نوعاً ما الأمر SUB ما عدا محتويات علم الباقي تطرح من السجل A .

: AND

مثال : ـ (AND A; AND (HL); AND 0 F H)

هذا الأمر يقوم بأداء العلاقة المنطقية AND بين السجل A وبين (بايت) معلومات محددة ويضع النتيجة في السجل A .

إن العلاقة المنطقية AND تعني المقارنة بين (بايتين)، (بت) بعد (بت) . فإذا كان محتوى كلا البتين «1» عندئذٍ تكون (بت) النتيجة المناظرة تساوي «1» وإلا تكون تساوي «0» .

وإليك المثال التالي يوضح هذه العلاقة :

لنفرض أن السجل A يحوي (0 A 7 H) (ستة عشر)، ونريد تنفيذ الأمر: (AND 0 FH) . فإليك طريقة المقارنة :

```
         10100111  (A7 hex, 167 decimal)
         00001111  (0F hex, 15 decimal)     بايت القناع
Result = 00000111  (7)
```

يستخدم هذا (التكنيك) عادة في عملية القناع (mask) أو الحجب ويكون لحذف جزء لا نريده من (البايت) ومعلومات التقنيع (masking) أو الحجب في المثال السابق هي «0 FH» لتغطي جزءاً من (بايت) المعلومات التي نريد حفظها. (لأنا نرى أن «0 FH» قد حجبت (البت) رقم 3 و 4 و 5 و 6 و 7 عن الظهور في (بايت) النتيجة) .

من الملاحظ أن عملية المقارنة المنطقية AND تعيد علم الباقي إلى وضعية الصفر دوماً «0» . ولهذا فإن الأمر AND A سوف يعيد علم الباقي إلى الصفر «0» ويدع السجل A كما كان على حاله قبل العملية : لذلك فإن هذا الأمر يمكن استخدامه في عملية تنظيف علم الباقي «C» بدون التسبب بأي إزعاج للسجل A .

: OR

مثال : ـ (OR 80 H, OR (HL), OR A)

يقوم هذا الأمر بأداء العلاقة المنطقية OR على السجل A ، ويضع النتيجة في السجل A .

وعملية المقارنة OR تعني الاختبار بين (بايتين)، (بت) بعد (بت) فإذا كان أحد أو كلا البتين يساوي «1» فإن بت النتيجة المناظرة عندئذٍ يساوي «1» وإلا يكون يساوي «0».

وإليك هذا المثال التالي يوضح هذه العلاقة :

لنفرض أن السجل A يحوي (1B hex) ولدينا الأمر OR 80 H فيعطينا :

البرنامج :

```
         00011011  (1BH, 27 decimal)
         10000000  (80H, 128 decimal)
Result = 10011011  (9BH, 155 decimal)
```

ان هذه الطريقة يمكن أن تكون مفيدة في الجمع على مستوى (البت) إلى (البايت): فإذا كان لدينا على سبيل المثال في السجل A القيمة بين (0 و الـ 9) فإن الأمر (OR 30H) سوف يترك في السجل A رمز الآسكي ASCII لهذا الرقم.

إن العلاقة المنطقية OR دوماً تقوم بتنظيف علم الباقي بعد تنفيذها وتؤثر بالأعلام الأخرى الموجودة طبقاً للنتيجة، ولهذا فإن الأمر (O R A) لا يغير قيمة السجل A بل ينظف علم الباقي.

: XOR

مثال : ـ (XOR A, XOR (HL), XOR 0 FH)

يقوم هذا الأمر بأداء العلاقة المنطقية XOR على السجل A، ويضع النتيجة في السجل A

والعلاقة (XOR) تعني المقارنة بين (بايتين) من المعلومات (بت) بعد (بت).

فإذا كان أحد البتات يساوي «1» والآخر يساوي «0» فعندئذٍ يكون بت النتيجة المناظرة يساوي «1» وإلا فإنه سوف يكون يساوي «0».

وهكذا إذا كان لدينا في السجل A القيمة 14 H فالأمر (XOR 17 H) يعطينا :

النتيجة

```
         00010100  (14H, 20 decimal)
         00010111  (17H, 23 decimal)
Result = 00000011  (3)
```

إن العلاقة XOR تعيد دوماً عَلَم الباقي إلى حالته الإبتدائية «0» وتؤثر ببقية الأعلام طبقاً للنتيجة. ومن المهم ذكره أن الأمر XOR A يجعل قيمة السجل A تساوي الصفر لذلك يستخدم طريقة ناجحة لتصفير السجل A أي جعل قيمة السجل A تساوي الصفر.

: CP

مثال : ـ (CP B, CP (HL), CP 9) •

يقوم هذا الأمر بطرح بايت معينة من المعلومات من القيمة الموجودة في السجل A «والتخلص من النتيجة». ولهذا فإن الأعلام فقط هي التي تتأثر من تنفيذ هذا الأمر. فإذا كان بايت الاختبار أكبر من القيمة التي في السجل A عندها يوضع في علم الباقي القيمة «1».

وإذا كان بايت الإختبار يحوي نفس القيمة التي في السجل A فإن علم الصفر «Z» سوف يوضع فيه القيمة «1».

أما إذا كان بايت الاختبار يساوي أو أقل من القيمة التي في السجل A فعندئذ يعاد علم الباقي إلى القيمة الابتدائية «0».

إن عَلَم الإشارة وعَلَم P/V سوف يحويان القيمة «0» أو القيمة «1» وذلك طبقاً للقيمة التي في السجل A.

مجموعة (الستة عشر) (بت) الحسابية والمنطقية (16-BIT)
The 16- BIT Arithmetic & Logic Group

كما في مجموعة الثماني (بتات) يوجد أيضاً أوامر بسيطة في هذه المجموعة وهي الأمر INC والأمر DEC.

وهذه الأوامر يمكن استخدامها في زيادة أو تنقيص السجلات الزوجية :

BC, DE, HL

وكذلك مع السجلات ذات (ستة عشر) (بت) (16-BIT) :

SP, IX, IY

ونلاحظ أن الأمرين INC وفي السجلات ذات (الستة عشر) (بت) لا يؤثران على حالة الأعلام (Flags) مطلقاً كما يؤثران في السجلات ذات ثمانية بتات (8-BIT)

الجدول التالي يرينا الأوامر: SBC, ADC, ADD الممكن تنفيذها على هذه السجلات : (الإشارة X تعني أنه يمكن) :-

الأوامر:

	This pair	with BC	DE	HL	SP	IX	IY
ADD	HL	x	x	x	x		
ADD	IX	x	x		x	x	
ADD	IY	x	x		x		x
ADC	HL	x	x	x	x		
SBC	HL	x	x	x	x		

لاحظ من خلال هذا الجدول أن الأمر SUB غير موجود لعدم إمكانية استخدامه ولكن علم الباقي دوماً مرتبط مع عملية الطرح (SBC) . وإذا كنت لا تريد إشراك علم الباقي في عملية الطرح، في هذه الحالة يمكن أن يحوي القيمة «1» ، استخدم الأمر O R A أولاً حتى تنظف علم الباقي (الأمر ORA ينظف علم الباقي راجع فقرة OR) .

إن وظائف كل من الأوامر SBC, ADC, ADD هي نفسها أوامر مجموعة ثمانية (بتات) (8-BIT) ما عدا أنها هنا تجري عملياتها على (ستة عشر) بت (16-BIT) .

الزحزحة والتدوير على الثماني بتات (8-BIT)

The 8-BIT Shifts and Routates

هناك بعض أوامر التي يمكن تجري على (بايت) معين من المعلومات كالزحزحة أو تدوير محتوياتها إلى اليمين أو إلى اليسار.

والبايت الذي يمكن أن تجرى عليها هذه العمليات هي :

A, B, C, D, E, H, L, (HL), (IX + d), (IY + d)

وهذه بعض الأوامر الممكن تنفيذها والتي تكون كما يلي :

RLC

(مثال : RLC B ; RLC (H))

يقوم هذا الأمر بترحيل محتويات (البت) ذي الرقم 0 إلى البت ذي الرقم 1 والبت ذي الرقم 1 إلى البت ذي الرقم 2 وهكذا .

البت ذي الرقم 7 ترحل إلى البت رقم 0 وإلى علم الباقي والمعلومات في هذه الحالة تكون قد دورت إلى اليسار وعلم الباقي يعكس محتويات البت ذي الرقم 7 .

لاحظ أنه مع أوامر السجل A يمكن أن نكتب (RLC A) أو (RLCA) : والأمر RLC A يختلف عن الأمر RLC A فهو يحتاج إلى بايت أمر واحدة أقل .

RRC :

مثال : (RRC (HL); RRC B)

يقوم هذا الأمر بترحيل محتويات البت ذي الرقم 7 إلى البت ذي الرقم 6 والبت رقم 6 إلى البت رقم 5 وهكذا . محتويات البت رقم 0 ترحل إلى علم الباقي وإلى البت رقم 7 وتكون المعلومات في هذه الحالة قد دورت إلى اليمين وعلم الباقي يعكس محتويات البت رقم 0 . ولاحظ أنه من أجل السجل A يمكن كتابة الأوامر حسب الشكل التالي :

RRC A أو RRC A والأمر الأخير يكون أقصر وأسرع في النقل من الأول .

RL :

مثال : (RL (HL), RL B)

يقوم هذا الأمر بترحيل محتويات البت رقم 0 إلى البت رقم 1 والبت رقم 1 إلى البت رقم 2 وهكذا، ومحتويات البت رقم 7 ترحل إلى علم الباقي ومحتويات علم الباقي ترحل إلى البت رقم 0 وهكذا فإن تسعة بتات تكون قد اشتركت في عملية التدوير اليسارية .

لاحظ أنه من أجل السجل A يمكن أن تكتب الأوامر حسب الشكل التالي : (RLA) بدلاً من أن تكتب (RL A) لأن الأمر RLA أقصر وأسرع .

RR :

مثال : (RR (HL), RR B)

يقوم هذا الأمر بترحيل محتويات علم الباقي إلى البت رقم 7 ومحتويات البت رقم 7 إلى البت رقم 6 وهكذا، محتويات البت رقم 0 ترحل إلى علم الباقي . وهكذا فإن تسعة بتات تكون قد اشتركت في عملية التدوير اليميني . أما بالنسبة للسجل A فإن الأوامر يمكن أن تكتب (RRA) بدلاً من أن تكتب (RR A) لأن الأمر RRA أقصر وأسرع من الأمرين .

SLA :

مثال : (SLA B, SLA (HL))

يقوم هذا الأمر بترحيل البت رقم 0 إلى البت رقم 1 والبت رقم 1 إلى البت رقم ٢ وهكذا. أما البت رقم 7 ترحل إلى علم الباقي، والقيمة «0» توضع في البت رقم 0، وهكذا تكون المعلومات قد أزيحت إلى اليسار.

SRA

مثال : (SRA B, SRA (HL))

يقوم هذا الأمر بترحيل البت رقم 7 إلى البت رقم 6 والبت رقم 6 إلى البت رقم 5 وهكذا. أما البت رقم 0 يرحل إلى علم الباقي . البت رقم 7 يعاد إملائها مرة ثانية

بقيمته الأصلية (هذا يكون لإشارة العمليات الرياضية لحفظ إشارة البت رقم 7) وهكذا تكون المعلومات قد أزيحت إلى اليمين حسابياً .

SRL

مثال : (SRL (HL), SRL B)

يقوم هذا الأمر بنقل البت رقم 7 إلى البت رقم 6 والبت رقم 6 إلى البت رقم 5 وهكذا. أما البت رقم 0 ينقل إلى علم الباقي والقيمة 0 توضع في البت رقم 7 . وبهذه العملية تكون قد زحزحنا المعلومات إلى اليمين.

التدوير الحسابي العشري :
Decimal Arithmetic Rotates

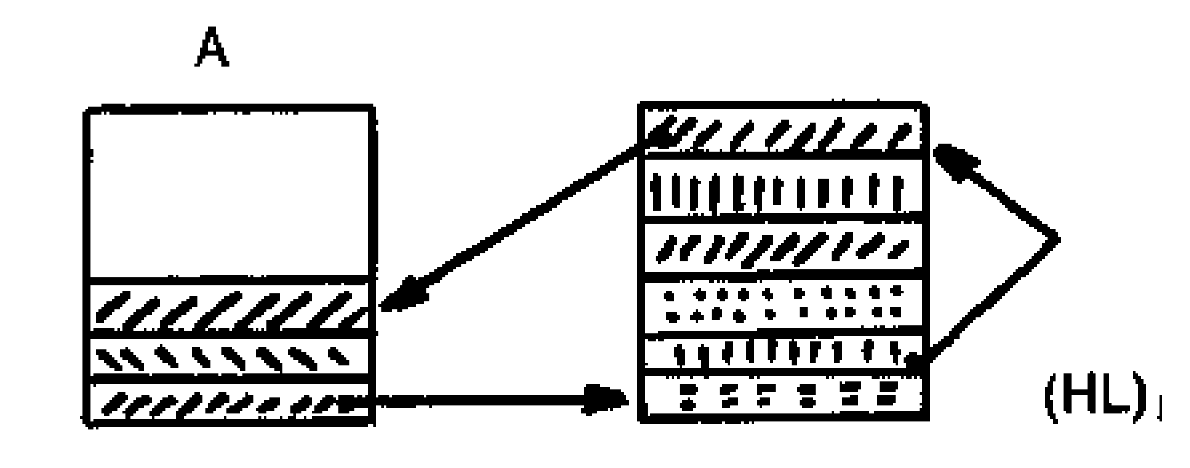
لقد أتينا الآن إلى إثنين من وظائف التدوير الخاصة جداً، وهي تستخدم عندما نستعمل الحساب العشري للترميز الثنائي . وإن كل من هذين الأمرين بين السجل A و (بايت) معلومات عنوانه موجود في السجل الزوجي (HL) .

فإليك هذين الأمرين :

RLD :

يقوم هذا الأمر بوضع البتات الأربعة السفلية للسجل A في مكان (البتات) الأربعة السفلية (لبايت) المعلومات الموجودة في العنوان (HL)، وهذه (البتات) الأربعة التي في أسفل (HL) توضع مكان (البتات) الأربعة التي في أعلى (HL) ، الأربعة التي في أعلى (HL) توضع مكان البتات الأربع السفلية للسجل A .

وهكذا تكون (البتات) الأربعة قد دورت وأما (البتات) الأربعة العليا في السجل A لم تتأثر من جراء هذه العملية .



RRD :

يقوم هذا الأمر بالعمل نفسه الذي يقوم به الأمر RLD ، ولكن بالإتجاه الآخر. فإن (البتات) الأربعة السفلية للسجل A ترحل إلى مكان (البتات) الأربعة العليا لبايت المعلومات الموجودة في العنوان (HL) ، و(البتات) الأربعة العليا ترحل إلى مكان (البتات) الأربعة السفلية للعنوان (HL) وهذه (البتات) الأربعة السفلية للعنوان (HL) ترحل إلى مكان (البتات) الأربعة السفلية للسجل A . ونلاحظ من خلال هذا الأمر أن (البتات) الأربعة العليا في السجل A لم تتأثر نتيجة هذه العملية .

معالجة البت :

BIT MANIPULATION

في أغلب الأحيان نريد اختبار (بت) محدد في (بايت) معلومات حتى نرى فيما إذا كان «0» أو «1» . وكذلك أيضاً من المفيد أن يكون لدينا القدرة بوضع القيمة «1» أو في «0» في بت محددة .

المعالج Z 80 يسمح لك القيام بهذه العملية .

يوجد هناك ثلاثة أوامر متاحة للقيام بهذه العملية وهي :

يقوم باختبار البت b في الموقع «1» :BIT b, 1

يضع القيمة «1» في (البت) الذي في الموقع «1» :SET b, 1

يضع القيمة «0» في (البت) الذي في الموقع «1» :RES b, 1

ـ البت b يمكن أن تكون بالطبع أي بت من (0 إلى 7) الصفر إلى السبعة .

(تذكر أن البت رقم 7 هو البت الأكثر أهمية وأن البت «0» هو البت الأقل أهمية) .

ـ الموقع «1» يمكن أن يكون في سجل من السجلات التالية :

A, B, C, D, E, H, L, (HL), (IX + d), (IY+ d)

وهكذا هناك ثلاثة أوامر أساسية كل واحد من هذه الأوامر يستطيع أن يعمل على بت واحد من أصل ثمانية بتات وبعشرة مواقع مختلفة ـ فالمجموع حوالي 240 أمر للكل.

وإليك الآن أمثلة نموذجية على هذه الأوامر الثلاثة :

BIT 3,B

يقوم الأمر باختبار فيما إذا كانت البت 3 في السجل B هي «0» أو «1» فإذا كانت «0» فإن عَلَم الصفر يوضع على «1» وهكذا فإن هذا الاختبار اللاحق للصفر سوف ينتج . لنرى هذه الجزء من البرنامج :

```
BIT 3, B
JPZ, WAS ZERO
```

سوف يتم القفز إلى الجزء من البرنامج المعنون بـ «WAS ZERO» إذا كان (البت) 3 في السجل B قيمته صفراً «0» وإلا فإن المعالجة سوف تتابع بالأمر الذي يلي .

نلاحظ أنه بينما يكون علم الصفر إما «1» أو «0» بالتحديد بعد تنفيذ أوامر (البت) (BIT) فإن علم الإشارة «S» والعلم «P/V» ربما يتأثران أو ربما لايتأثران وإن المعلومات التي بداخلها تكون غير متعلقة بهذه الحالة وغير قابلة للاختبار.

علم الباقي لا يتأثر بهذه العملية فهو سيحوي القيمة السابقة التي وجدت فيه .

SET 7, (HL)

يقوم هذا الأمر بجعل (البت) رقم 7 من (بايت) المعلومات المحددة بالعنوان الموجود في السجل الزوجي HL مساوية للواحد «1» .

RES 5, (IX,3)

هذا الأمر يعمل على (بايت) المعلومات المحددة بالعنوان الموجود في السجل IX مضافاً إليه إزاحة مقدارها -3- ويجعل البت رقم 5 مساوية للصفر.

وهكذا فإن كان السجل IX يحوي القيمة «8000 H» فعندئذٍ فإن (بايت) المعلومات في العنوان «800 3H» سوف يتحول البت رقم 5 فيها إلى الصفر «0» .

إن وظائف معالجة (البت) تستطيع أن تثبت أنها مفيدة جداً في بعض أنواع البرامج . لنعطي مثالاً واحداً فقط على هذه البرامج الواسعة ، فمثلاً في برامج ألعاب المغامرات ربما يكون (بايت) معلومات واحدة تستخدم للإشارة إلى امكانية الخروج

من موقع معطى (الـ «0» يعني لا يوجد خروج) والواحد «1» يعني يمكن الخروج. (البت) رقم 7 يستطيع أن يمثل الشمال و(البت) رقم 6 يستطيع أن يمثل الشرق وهكذا، واربعة بتات يسارية تمثل الذهاب إلى فوق، تحت. واثنان لامكانية الخروج. فالاختبار يعني إذا أمكن الخروج أولاً في هذه الحالة ممكن وبسيط. فما هو إلا اختيار ما يناسب (البت): (تغيير حالة الخروج يكون بسيط فما هو إلا بوضع «1» أو «0» بهذه البت).

معالجات خاصة للسجلين F و A
SPECIAL A and F REGISTER MANIPULATION

هناك خمسة أوامر تعمل بشكل خاص على السجل A أو على عَلَم الباقي في السجل F، إليك هذه الأوامر على الشكل التالي:

DAA

يستخدم هذا الأمر بشكل خاص عند انجاز الحساب العشري للترميز الثنائي (BCD). ففي هذا الأخير كل أربعة (بتات) (4-BIT) تستخدم لتخزين رقم عشري واحد. وهكذا فإن (البايت) يستطيع تخزين رقمين عشريين (هذا يشير إلى الـ (BCD) المضغوطة). القيمة «11» إلى «15» العشرية يمكن أن تمثل بأربعة (بتات): ومهما يكن فإن في الـ BCD نريد فقط رقم عشري واحد في كل أربعة (بتات)، وهكذا فإن التمثيل الثنائي للرقم العشري «11» إلى «15» هو أقل مما نريد وغير مطلوب.

دعنا الآن نرى هذين المثلين: الأول نود من خلاله جمع الرقم العشري «22» إلى الرقم العشري «43» والبرنامج الذي يقوم بهذا الجمع العشري المرمز بشكل ثنائي هو حسب ما يلي:

برنامج:

```
LD  A,22H;22H = 0010 0010 binary,'22' in BCD
ADD A,43H;43H = 0100 0011 binary,'43' in BCD
```


كما نرى أن جمع القيم الثنائية هذه سوف يعطينا 0110 0101 وهذا ما يعادل في الـ BCD الرقم «65» وهذا ما نريده.

دعنا الآن نرى ما سيحدث إذا أردنا جمع الرقم العشري «26» مع الرقم العشري «17» نستخدم لذلك البرنامج الجزئي السابق، والتمثيل الثنائي لهذه الأرقام العشرية يكون كما يلي:

```
0010 0110   (26H)
0001 0111   (17H)
```

وبعد الجمع نجد:

```
and if we add these, we get:

0011 1101   (3DH)
```

في هذه النتيجة الـ «D» لا معنى لها كرقم عشري. وإن القارىء الصبور يعلم أين يجب أن يأتي الأمر DAA. لنقم بإضافة هذا الأمر بعد الأمر «ADD A» في البرنامج السابق، فما هو إلا تعديل للنتيجة العشرية التي في السجل A.

وهكذا نرى أنه في المثال الأول الأمر «DAA» لا يقوم بشيء لأن النتيجة صحيحة، أما في المثال الثاني رأينا أنه كيف بعض الأشياء وقد تمت بشكل خاطىء بالنسبة (للبتات) الأربعة الدنيا «3 D» لناتج الجمع (وهذا يعتمد على طبيعة العملية إذا كنا نريد أن نطرح أو أن نجمع) وهذا يحتاج إلى تعديل النتيجة حسب الحالة.

ففي المثال الثاني ندع السجل A يحوي القيمة الثنائية (0100 0011) -(«43 H») أو 43 في النظام العشري ـ في هذا المثال بالتحديد تم انجاز النتيجة بجمع 6 أيضاً إلى (البتات) الأربعة الدنيا (ولكن لا تقلق حول هذا). يكفي أن تعلم أن هذا جعل التعديل صحيح.

ولكن ما يجب أن تعلمه على أي حال أنك عندما تريد اخراج نتيجة صحيحة فما عليك إلا استخدام الأمر DAA فهو يقوم باستعمال الأعلام حسب حاجته، وبالتالي بعد استخدام الأمر DAA فإن جميع الأعلام تتأثر بطريقة ما.

CPL

يقوم هذا الأمر بتكملة الموجودة في السجل A مهما تكن: وذلك بتبديل كل صفر بواحد «1» وكل «1» يصبح صفراً «0». وهكذا فإذا كان السجل A يحوي القيمة الثنائية (00101100) فبعد الأمر CPL سوف يحوي القيمة الثنائية (11010011) تدعى هذه الطريقة بالتكملة الأحادية للرقم وهي طريقة لتمثيل القيم الموجبة والسالبة.

على سبيل المثال: الرقم «5» في النظام الثنائي يمثل بـ (00000101).

ومن ناحية أخرى الرقم «-5» يمكن أن يمثل بطريقة التكميل الأحادي بـ (11111010).

لاحظ أن (البت) رقم 7 أصبح الآن «1» لتمثل القيمة السالبة.

أما في هذه الحالة فإن الأعلام التي يمكن أن يجرى عليها الاختبار لا تتأثر.

NEG

في هذا الأمر محتويات السجل A تطرح من الصفر والنتيجة تعاد مرة ثانية إلى السجل A وهذا ما يدعى بالتكملة الثنائية للرقم.

في طريقة التمثيل بالتكملة الثنائية، القيم الموجبة تمثل كما في طريقة التكملة الأحادية تماماً. ففي طريقة النظام الثنائي يستدل على القيمة بأنها موجبة أو سالبة من (البت) السابقة 7-BIT (0 = موجبة، 1 = سالبة).

فالأرقام السالبة مهما تكن تمثل كما في طريقة التكملة الأحادية للقيم مع إضافة واحد عليها.

وبهذا فإن الرقم «-5» يمثل بطريقة التكملة الثنائية ويكون (11111011).

لماذا وضعنا أنفسنا في كل هذه الحيرة؟ نستطيع القول أن التكملة الثنائية تجعل الإشارة الحسابية أسهل بالنسبة للحاسب ليقوم باستعمالها.

لنرى عملية الجمع هذه بين الرقم «3» و الرقم «-5».

000011 (+3)

1111011 (-5) حيث

بعد تمثيل الرقم -5 بطريقة التكملة الثنائية نحصل على: 11111110

في هذه النتيجة (البت) رقم 7 يدلنا على أن النتيجة سالبة.

لنأخذ التكملة الثنائية للرقم 11111110 فنحصل على 00000010 (تذكر أن التكملة الثنائية هي عبارة عن التكملة الأحادية للرقم 11111110 والتي هي 00000010 مضافاً إليها واحداً «1»)، وبهذا فإن القيمة تكون «2» والإشارة السالبة، النتيجة -2.

الأمر NEG في المعالج Z 80 يحصل على التكملة الثنائية للقيمة التي في السجل A ثم يضع النتيجة في السجل A أيضاً، وبهذا نتجنب الحيرة والمضايقة من عمل التكملة الأحادية CPL ثم إضافة واحد (ADD A,1).

إن هذا الوصف قليل جداً بالنسبة للمبادىء التي وراء التكملة الأحادية والثنائية الحسابية ولكن نعتقد أن هذا كافي للدخول إلى لغة الآلة وإعطاء فكرة عنها.

- يجب الملاحظة أن جميع الأعلام ربما تتأثر بالأمر «NEG».

: CCF

هذا الأمر يكمل عَلَم الباقي في السجل F. فإذا كان عَلَم الباقي «0» فإن CCF عندئذٍ تجعله «1» وإذا كان عَلَم الباقي «1» تجعله «0».

: SCF

هذا الأمر يجعل عَلَم الباقي مساوياً للواحد «1» أي يضع فيه «1»

ليس هناك أمر يقوم بتصفير عَلَم الباقي لتنظيفه ولكن كما ذكرنا سابقاً أن كلاً من AND و OR يقومان بذلك بدون أن يؤثر ذلك على أي شيء آخر.

XOR تقوم بتنظيف عَلَم الباقي أيضاً ولكن تقوم بتنظيف السجل A أيضاً معه أي تجعله «0» وهذا مما يؤدي إلى جعل علم الصفر مساوي للـ «0» أيضاً ومن ثم علم الإشارة سوف يتأثر أيضاً (البت 7).

إن القارىء المتيقظ يرى أن هناك عدة طرق يمكن اختيارها لتنظيف علم الباقي فيمكن أن يضع أولاً الأمر (SCF) وبعد ذلك يقوم بتكملة علم الباقي (CCF). ولكن هذه الطريقة تأخذ (بايتين) من الأوامر المرمزة، بينما ORA تأخذ فقط (بايت) واحداً. فهي ليست جيدة بشكل تام لكنها تعمل على أية حال.

مقارنات الكتلة :

BLOCK COMPARISONS

آخر أوامر المعالجة والاختبار التي تقوم بالفحص هي مقارنات الكتلة .
يمكن اعتبار هذه الأوامر مشابهة إلى حد ما إلى أوامر نقل الكتلة التي تم مناقشتها فيما سبق .

هذه الأوامر تسمح بالبحث في كتلة كاملة من المعلومات لإيجاد (البايت) الذي له مثيل في السجل A . كما هو الحال في أوامر نقل الكتلة ، فهي لا تحتاج منك إلا أن تجهز السجلات أولاً : تضع في السجل HL عنوان البداية للمنطقة التي سوف يتم فيها البحث ، تضع في السجل BC عدد (البايتات) التي ستبحث ، ونضع في السجل A (بايت) المعلومات التي نريد البحث عنها . وإليك هذه الأوامر:
أوامر:

```
CPI    Increment HL
       Decrement BC

CPD    Decrement HL
       Decrement BC

CPIR   Increment HL
       Decrement BC
       Continue until BC=0 or A=(HL)

CPDR   Decrement HL
       Decrement BC
       Continue until BC=0 or A=(HL)
```

وكما في نقل الكتلة من المعلومات للأمرين CPI و CPD يسمحان بعمليات أخرى تكون هي المسؤولة داخل حلقة البحث هذه ، وعندما يتم العثور على النظير فإن عَلَم الصفر يصبح به واحداً (SET) . وعندما السجل BC يصل إلى الصفر فإن العلم P/V يصبح صفراً «0» (RESET) .

إن الأمرين CPIR والـ CPDR يقومان بالبحث داخل الكتلة حتى يتم العثور على النظير أو يصبح السجل BC مساوياً الصفر . وعندما يتم ايجاد النظير الذي يتم البحث عنه فإن السجل الزوجي HL يشير إلى (البايت) المناظر (المشابه) الذي في كتلة المعلومات .

تحويل مسار برنامج ينفذ بشكل تسلسلي :

Re- Routing Program runing Sequence

لقد أتينا الآن إلى الأوامر التي تسمح لك بكبح أو تحويل تعليمات البرنامج الذي لديك ، والتي يوجد في لغة (البيزيك) ما يقابلها مثل GOSUB, GOTO ، وطبعاً RETURN أما في لغة الآلة فإن لديك مجالاً أكبر لمثل هذه الأوامر .

القفز والقفز النسبي :

Jumps and Relative Jmps

ان تعليمة GOTO في لغة (البيزيك) يقابلها تعليمة القفز (JP) أو القفز النسبي (JR) . أما بالنسبة للقفز العادي المستقيم فهو تماماً مثل GOTO فهي تقفز من رقم سطر إلى رقم سطر آخر مباشرة ، وصيغة الأمر تكون على الشكل التالي :

JP Label or JP address

لأن الـ Label يمثل اسم فقرة محددة من البرنامج تكون قد سميتها نفسها في برنامج لغة التجميع . أو تكون قد حددت بمعادلة .
يمكن للقفز أن يكون مشروطاً وذلك باختبار لأحد الأعلام ، فإذا تحقق الاختبار يتم القفز . والصيغة لهذا الأمر هي :

JP CC, Label or JP CC, adress

لأن CC تمثل أي شرط من شروط الأعلام التي يمكن اختبارها.
(راجع قسم الأعلام -e.g. NZ, Z, NC, C, PO, PE, P, M)

وهكذا فهذه الأنواع من الأوامر يمكن أن تكون على الشكل التالي: JP NZ, ENDGAME والتي تعني إذا كان علم الصفر Z» لا يساوي «0» الصفر كنتيجة لعملية سابقة عندئذٍ اذهب مع المعالجة من العنوان المسمى بـ «ENDGAME».

أما بالنسبة للقفز النسبي فهو يحتاج إلى قليل من الشرح فرموز الأوامر للقفز النسبي أقصر من أوامر القفز المباشر أو المستقيم. فالعنوان الذي يزود القفز هو عنوان نسبي للعنوان الحالي ويعطى مع مقدار زحزحة معين: وبناء على ذلك فالعنوان الحقيقي لا يوجد في رمز الأمر بذاته.

وإن لم يوجد أياً من العناوين في داخل الروتين بذاته فقد أشير إليه بشكل مباشر فإن (الروتين) يمكن أن يتوضع في أي مكان من الذاكرة. وهذا ما يدعى (بالروتين) الذي يمكن إعادة توضعه من جديد. فكثيراً من (المبرمجين) يقومون بكتابة (روتينات) فرعية صغيرة (لتقوم بعمل محدد) بشكل يقبل اعادة التوضع من جديد، وهكذا فإن باستطاعتهم إضافة هذا (الروتين) إلى أي برنامج رئيسي يقومون بتحضيره. فكل ما يحتاجونه هو عنوان البداية لهذا (الروتين) والذي يكون عادة عن طريق اسم فقرة (Label) تكون في البرنامج.

صيغة القفز النسبي تكون على الشكل التالي:

JR Label or JR, SC, Label

لأن SC تمثل اختباراً شرطياً. لا كما في القفز العادي يمكن اختبار أي من الأعلام، ففي القفز النسبي فقط علم الصفر «Z» وعلم الباقي «C» يمكن اجراء الاختبار عليهم:

Z, NZ, C, NC

وهكذا فليس باستطاعتك كتابة على سبيل المثال: JR, M, LABEL

ويمكن للقفز النسبي أن يكون إما إلى الأمام أو إلى الوراء. فمقدار الزحزحة يكون بالتكميل الثنائي ويضاف إلى عداد البرنامج مع إضافة (2). فإذا قمت

بالعمل حسب ما سبق فإنك سوف تجد أن القفز النسبي قد حصل داخل العنوانين 126- و 129 + BYTES) ابتدىء من عنوان أول (بايت) من الأمر JR.

ولحسن الحظ أن المجمع يقوم بحساب مقدار الزحزحة تماماً عندما تقوم بتوليد لغة الآلة فلا داعي للقلق حول هذا الموضوع.

القفز الخاص:

Special Jumps

هناك أربعة أنواع من القفز يمكنك استخدامها في لغة الآلة. ثلاثة منها تسمح لك بالقفز إلى عنوان محدد في السجلات. مثال ذلك:

JP (HL)

JP (IX)

JP (IY)

وهذه الأوامر الثلاثة مفيدة جداً عندما تستخدم جدول القفز. حيث يمكن استخدام على سبيل المثال جدول معلومات (لبنود)، كل بند مؤلف من ثلاثة (بايتات).

ـ البايت الأول من كل بند يكون منتقي للائحة الخيارات، و(البايتين) المتبقين يكونان عنوان (روتين) العمل للائحة خيارات البند. (تذكر أن في كل أمر (بايت) أدنى وبايت أعلى)، يقوم منتقي لائحة الخيارات (mem Selector) بالبحث خلال جدول المعلومات حتى تحصل المطابقة (يقوم بالقفز فوق (البايتين) التاليتين في البند إذا لم يكن هناك تطابق في المعلومات).

عند استخدام السجل HL للإشارة إلى (بايت) التطابق يكون الأمر سهلاً بزيادة السجل INC HL- HL- (يشير إلى (البايت) الأدنى لعنوان العمل) LD E, (HL)- نقوم بالتقاط (البايت) الأدنى في السجل INC HL- E- وبزيادة السجل HL (يشير إلى (البايت) الأعلى إلى عنوان العمل).

LD D, (HL)- ثم إلتقاطها بـ E X DE, HL;- ثم نضع العنوان في السجل HL ثم ننفذ الأمر JP (HL)

هـذه الطريقـة هي واحـدة من عدة طرق، فهنـاك عدة طرق تستطيع التقاط العنوان (للروتين) المطلوب. وتعتبر هذه الطريقة بدائية ولكنها فقط لتوضيح هذه النقطة.

النوع الرابع من القفز يشابه إلى حد ما الأمر FOR-NEXT في لغة (البيزيك)، ولكنه يعتبر نوعاً من أنواع القفز النسبي ويأخذ الشكل التالي:

DJNZ Labe'

يستخـدم مع هذه التعليمـة السجـل B كعداد، ولهذا يجب عليك أن تضع فيه قيمة تساوي إلى عدد المرات التي تريد تكرارها لعملية ما.

في بداية الحلقـة التكـرارية تضع اسم أوعنوان لهذه الحلقة التكرارية (Label) ومع تنفيذ الأمر DJNZ يتناقص السجل B فإذا لم يكون محتوى هذا السجل صفراً «0» فالقفز سوف يتم من جديد إلى عنوان اسم الحلقة التكرارية.

ولهـذا يعتبر هذا النـوع من القفـز بالقفـز النسبي. وهكذا فعنوان اسم الحلقة التكـراريـة يجب أن يكـون ضمن 126- و 129 + (BYTES) من عنـوان الأمـر DJNZ (البرنامج المجمع يقوم بحساب مقدار الزحزحة المناسب).

يمكن القفـز خارج هذه الحلقـة التكـراريـة في أي وقت إذا تحقق اختبار فرعي موجود ضمن هذه الحلقة. فالسجل B سوف يحتوي في هذه الحالة عدد المرات التي لم تنفذ عند تحقق الاختبار الفرعي ـ والتي تكون معلومة مفيدة في بعض الأحيان.

الاستدعاء:
CALLS

يعتبر الأمر (CALL) مشابهاً تماماً للأمر GOSUB في لغة (البيزيك) وكذلك مثل أمر القفز -JP- ويمكن أن يكون غير مشروط:

CALL Label or CALL adress

أو يمكن أن يكون مع شرط:

CALL CC, Label or CALL CC, address

حيث CC تمثل احدى اختبارات الأعلام كما هو الحال في أمر القفز المشروط.

عنـدمـا ينفـذ الأمـر CALL فإن عنوان عداد الـبرنـامـج للأمر التالي يوضع في المكـدس (STACK)، مجهـز للتنفيـذ بعـد إلتقاء البرنامج بالأمر (RETURN). ولقد بحثنا هذا الموضوع عندما استعرضنا السجلات للمعالج Z 80.

ولهـذا يجب عليك أن تتأكـد أن المكدس ما زال يحوي عنوان الـ RETURN في الأعلى عندما الأمر -RETURN- قد نفذ (وإنها لكارثة حقيقية إذا لم تفعل ذلك).

إعادة التخزين مرة ثانية:
RESTORE

هناك نوع آخر خاص من أمر الإستدعاء (CALL)، يدعى بـ RST والتي تعني Restore أي إعادة التخزين مرة ثانية، وله الشكل التالي:

RST a

حيث a هي عبارة عن أحد عناصر المجموعة التالية:

00 H, 08 H, 10 H, 18 H, 20 H, 28 H, 30 H أو 38 H

عندما يقوم البرنامج بالاصطدام بالأمر RST، فإن عنوان عداد البرنامج يوضع في المكدس (كما في الأمر CALL) ويتم القفز إلى عنوان محدد.

سوف تلاحـظ أن كل العنـاوين المهمـة موجودة داخل منطقة الـ ROM. ولهذا على سبيـل المثـال فإن الأمـر -RST 00H- سوف يعطيك بدايـة جديـدة وكأنك قمت بضغط الزر (Rest) أي كأنك أطفأت الجهاز وقمت بتشغيله مرة ثانية.

أمـا العنـاوين الأخـرى فهي تزودك بالقـرف إلى (روتينات) معينة تستخدم في (بيزيك) الـ MSX، (الوصول إلى الحرف التالي في سطر البيزيك للنص الموجود على الشاشة) وهكذا.

الإرتداد ـ أو العودة:
Returns

يراقب هذا الأمـر الـروتـين الفرعي، تمامـاً كما في لغـة البيزيك RETURN وله الشكل التالي:

RET or RET CC

حيث CC هي أحد اختبارات الأعلام . كما في أمر القفز (JP) . وأمر الاستدعاء (CALL) .

هناك أمرين خاصين من أوامر العودة (RETURN) .

الأول هو (RET I) أي العودة من حيث تم التوقف أو الإنقطاع . والتي يجب دوماً أن تكون مسبوقة بالأمر EI (Enable Interrup T) (قابل للإنقطاع) .

الثاني وهو (RET N) ، والذي يقوم بتزويد العودة من الانقطاع الذي ليس قابلاً للحجب ويقوم أيضاً بتصفير عَلَم الانقطاع للمعالج Z 80 من شرط كان لديه قبل حدوث الانقطاع الذي لا يكون قابلاً للحجب .

أوامر الإدخال والإخراج :

Input/ Output Commands

هناك عدد من الأوامر المتاحة للإدخال أو الإخراج من وإلى الوحدات المحيطية في العديد من الطرق معظم هذه الأوامر تشبه أوامر نقل الكتلة (نقل المعلومات من مكان إلى آخر) . ولهذا فهي تسمح بنقل كتلة من المعلومات إما بشكل أوتوماتيكي أو عن طريق حلقة تكرارية يقوم بإنجازها أوامر أخرى .

وإليك أوامر الإدخال والإخراج هذه :

أوامر الإخراج	أوامر الإدخال
Output commands	Input commands
OUTI	INI
OTIR	INIR
OUTD	IND
OTDR	INDR

بالنسبة لأوامر الإدخال يقرء عنوان الوحدة المحيطية عن طريق السجل «C» وتشحن المعلومات إلى عنوان مشار اليه بوساطة السجل الزوجي HL . عندئذٍ السجل B يتناقص والسجل الزوجي HL يتزايد (INI , INIR) أو يتناقص (IND, INDR) .

أما بالنسبة لأوامر الإخراج ، فإن الأمر معكوس ، حيث أن محتويات العنوان المشار اليه بالسجل الزوجي HL يكون هو إخراج إلى وحدة محيطية معنونة بواسطة السجل C ، أما السجل B يتناقص والسجل الزوجي HL يتزايد أو يتناقص بعد كل عملية نقل .

إن أوامر الإدخال والإخراج تنتهي بـ «R» حيث أن التنفيذ يستمر حتى يصبح محتوى السجل B مساوياً للصفر B = 0 .

هناك أيضاً أربع أوامر للإدخال والإخراج يمكن استخدامها وهي :

أوامر الإخراج	أوامر الإدخال
Output commands	Input commands
OUT (p),A	IN A,(p)
OUT (C),r	IN r,(C)

لأن الأمر IN A, (P) يقوم بشحن السجل A (ببايت) من المعلومات مقروءة من بوابة وحدة محيطية «P» . وكذلك بالمثل الأمر OUT (P),A ، يقوم بإخراج (بايت) معلومات من السجل A إلى بوابة وحدة محيطية «P» .

أما بالنسبة للأمرين IN r, (C) والأمر OUT (C),r فهما يقومان بالمهمة نفسها، ما عدا وحدة البوابة تكون معنونة بالسجل C ، أما بالنسبة للسجل الموصوف «r» فيمكن أن يكون أي سجل من السجلات التالية :

A, B, C, D, E, H, L

التحكم في النظام :

System- Controls

هذه الأوامر تستخدم عادة للتحكم في نظام المعالج Z 80 :

: NOP

يعني هذا الأمر «لا يوجد عملية» (NO operation) أي أنه لا يقوم بأي شيء .

يستفاد من هذا الأمر والأمر الذي سيأتي بعد قليل (HALT) عندما تكتب برامج بلغة التجميع ليزودنا بالتوقف في النقطة المناسبة التي يمكن أن نختارها . ويمكن أن يستخدم أيضاً في عملية التأخير لفترة قصيرة جداً .

: HALT

هذا الأمر يوقف تماماً عمليات المعالج Z 80 ، حتى تصل إشارة انقطاع لهذا التوقف أو حتى تنفذ عملية تجهيز المعالج من جديد (Reset) أي إطفاء الجهاز وتشغيله مرة ثانية .

: DI, EI

ويعني هذان الأمران أنه لا يمكن أو يمكن إجراء عملية الانقطاع لسير عمل البرنامج . ولقد نوقشت عملية الانقطاع في فصل سجلات المعالج Z 80 .

IM 0, 1 or 2

يقوم الأمر IM بتجهيز المعالج Z 80 بنوع خاص من أنواع الانقطاع . (راجع مناقشة الانقطاعات في فصل سجلات المعالج Z 80) .

تعليمات غير متعلقة بالمعالج Z 80

NON Z 80 COMMANDS (pseudo OPS)

إذا كنت تستخدم المجمع ، فإنك ستجد أوامر أخرى يمكن استخدامها والتي تعتبر جوهرية في كتابة البرامج بلغة التجميع .

تستخدم هذه الأوامر فقط من قبل المجمع لأنك تخبره بما سوف يقوم به من عمل . حجز مكان للمعلومات ، التجميع عند عنوان معين وهكذا .

هذه الأوامر لا يمكن ترجمتها إلى رموز أوامر المعالج Z 80 ، وليس من الطبيعي ظهور هذه الأوامر في قائمة فك لغة الآلة إلى لغة التجميع .

وللمزيد من المعلومات عن هذه الأوامر أرجو أن تراجع كتيب التشغيل للمجمع الخاص بك فإنك سوف تجد تفصيلاً وافراً حول هذه الأوامر .

الفصل الثالث

استخدام المعالج ZEN

Using ZEN Assembler

هذا الفصل سوف يتعامل مع بداية كتابة برنامج لغة الآلة الخاص بك مستخدماً برنامج المجمع والمحرر مثل المجمع ZEN الذي يعتبر معروفاً ومتاحاً بشكل واسع بالنسبة لجميع الحواسب المنزلية التي تعمل بنظام الـ MSX

إن أي اختلاف في إدخال البرنامج بين المجمع ZEN والمجمعات الأخرى يعتبر بسيطاً جداً حيث أن المبدأ بالنسبة لجميع المجمعات هو نفسه .

فإذا كنت تعرف أو لديك فكرة عن طرق إدخال الأسطر في المجمع فإن بإمكانك القفز عن بعض أجزاء هذا الفصل لأنه سوف يكون واضحاً بالنسبة لك .

لأننا سوف نبدأ من (شحن) المجمع ووصف بعض الأخطاء التي يمكن أن يقع بها المستخدم لأول مرة . إن البرنامج الأول الذي سوف نقوم بإدخاله يعتبر بسيطاً وسهلاً في الطباعة لأنه لا يتجاوز طول سطر الشاشة الواحد ، فهو ليس بالبرنامج الممتع لكنه جيد وقصير لعرض كيفية إدخال الأسطر .

إن قسم الـ ROM من الذاكرة (العناوين بين 0000 والـ 8000 Hex) في نظام الـ MSX لا يحتوي فقط على مفسر لغة (البيزيك) ولكن تحوي أيضاً على (روتينات) تقوم بتنفيذ بعض الأعمال مثل طباعة الحروف على الشاشة ، طباعة سطر جديد ، تداول الساعة ، استخدام الشريحة PSG ، قراءة البرنامج على الشريط ، التحقق من البرنامج وحفظه على وحدة محيطية . . الخ .

وبشكل فعال وواضح تقوم هذه (الروتينات) بالعمل أثناء تنفيذ البرامج المكتوبة بلغة (البيزيك) أو بلغة الآلة لطباعة رسائل على الشاشة حيث تمّ استدعاء هذه

الـرسـائـل عن طريق (روتينـات) تكـون مكتوبة في الـ ROM فتوفر على نفسك كتابة (روتين) في برنامجك الخاص ليقوم بنفس العمل.

ـ يشحن المجمع ZEN داخل منطقة الـ RAM عند العنوان A000 hex ولهذا نقوم بإدخـال الأمـر CLEAR 200, & H 9 FFF قبـل (BLOAD «ZEN», R) إدخال أو (شحن) المجمع «ZEN» .

وبعد الانتهاء سيظهر على الشاشة : ZEN >

الآن حاول إدخال الأوامر الموجودة تحت العمود TO ENTER حسب ما هو وارد بالضبط مع مراعاة الفراغ الموجود أيضاً عند الإدخال.

وبعد الانتهاء من إدخال كل سطر يضغط المفتاح «RETURN» .

هناك خطأ في البرنامج قد أدخل قصداً للتدريب وسوف نقوم بتعديله بعد قليل تذكـر أن أي استـدعـاء أو قفـز إلى عناوين بين 000 h والـ 8000 H تكون (لروتينات) موجودة ضمن منطقة الـ ROM والوظائف الخاصة بهم سوف نأتي على وصفها.

برنامج

```
                TO
DISPLAYED       ENTER

ZEN >           E
   1            LOOP:EQU  0A003H
   2            CALL  0849H
   3            LD A,"A"
   4            NEXT:CALL00A2H
   5            INC A
   6            CP "Z"+1
   7            JR NZ,NEXT
   8            LD A,0DH
   9            CALL  00A2H
  10            LD A,0AH
  11            CALL  00A2H
  12            JP LOOP
  13            END
  14
ZEN >
```


عنـد نهايـة الـبرنـامـج يجب إدخـال «END» بسطر مستقل، ولإنهاء الإدخال والعودة ثانية إلى مستوى الأمر ZEN > يجب إدخال نقطة «.» بسطر مستقل أيضاً.

لنقم الآن بتحليل ما قد تم إدخاله.

السطر الأول من البرنامج :

هو عبارة عن سطر توازن ليقوم ببساطة باخبار المجمع أن عنوان الحلقة مساوي لـ A00 3 H وهـو العنـوان الـذي نرغب أن يقفز البرنامج إليه في نهاية المعالجة كما أشرنا إلى ذلك في السطر رقم 12 لأنا قمنا بإدخال الأمر JP LOOP ، فإننا لا نحتاج في هذه الحـالـة إلى تحديد العنوان الذي يجب أن يتم القفز إليه وأن المجمع يعرف تماماً عنوان الحلقة (LOOP) التي سيقفز إليها.

إن الغـرض من هذه المـوازنـة في السطـر الأول، إن أردنـا تعـديـل العنـوان في المستقبل لسبب ما، فإننا لا نحتاج إلى طباعة البرنامج كله ونقوم بتعديل كل سطر من أسطـر الـبرنـامـج الـذي يحوي هذا العنوان، أما في طريقتنا هذه فكل ما نحتاجه هو تعديل السطر الأول فقط إلى العنوان الذي نريده والمجمع يقوم بالعمل بعد ذلك.

إن هذا العنـوان هو نقطـة دخـول البـدايـة السـاخنة للمجمع ZEN . فعندما ينتهي هذا البرنامج القصير من التنفيذ فإننا نحتاج إلى إخبار الحاسب إلى أين سيقوم بالقفز وإن الحلقة الرئيسية للمجمع ZEN تبدو في مكان مناسب وحيد في هذه المرحلة، حيث أننا لا نريد لهذا البرنامج أن ينفذ بشكل طائش في الذاكرة.

ملاحظة :

إن كل رقم (ستة عشر) يبدأ بحرف من الـ (A-F) كما في مثالنا السابق يجب أن يسبق بالـرقم صفـر «0» كما رأينـا في السطر رقم واحد. وإلا فإن المجمع سيخلط بينه وبين عنوان الفقرة الذي يبدأ عادة بحرف أيضاً.

ومن ناحية أخرى فإن علامة الوقف (:) يجب أن تدخل بين عنوان الفقرة وبين الأحرف EQU . كما هو واضح في السطر الأول.

السطر الثاني من البرنامج :

يقوم باستدعاء (روتين) من منطقة (الروم) (ROM) موجود في العنوان 0849 H وهـو يقـوم بتنظيف شاشـة الحاسب ثم يعود إلى برنامجنا ليقوم بتنفيذ الأمر الذي يلي.

وهـذا يشـابه تماماً تعليمة GOSUB في لغة (البيزيك) وفي هذه الحالة البرنامج الفرعي موجود في منطقة الروم (ROM) وكل ما نحتاجه هو استدعاؤه فقط.

السطر الثالث من البرنامج :

يقوم (بشحن) السجل A بقيمة هي عبارة عن الحرف «A» .

إن المجمـع ZEN يمكن التعـامـل معـه بشكـل مرن لأنه يسمح لك بإدخال ما تريـده ضمن قوسـين علويـين ويقـوم بعـد ذلـك بتحويله إلى ما يعادله من القيمة في النظام (الستة عشر).

في الحقيقة هذا السطر سوف يؤدي نفس المهمة لو أننا أدخلنا LD A, 41 H لأنه يمكن أن تجمـع و(تشحن) إلى الـذاكـرة عن طريق المجمـع ZEN على أي حال 41 Hex هو قيمة الحرف «A» في نظام (الآسكي) ASCII (الستة عشر)، أو يمكننا إدخال LD A,65- وهي عبـارة عن قيمـة الحـرف «A» في نظـام (الآسكي) ASCII العشري، وإسقاط السابقة H وهي تعني بالنسبة للمجمع ZEN أن تلك القيمة عشرية والمجمع ZEN يلزمه تحويلها إلى (ستة عشر).

السطر الرابع من البرنامج :

يحوي هذا السطـر العنـوان NEXT الـذي سنقفـز إليه عائدين من جديد لمتابعة طبـاعـة الأحـرف. والعنوان NEXT متبوع بعد علامة التوقف : بالأمر CALL 00 A 2H وهو عبارة عن عنوان آخر في منطقة (الروم) ROM (لروتين) فرعي والذي يقوم بطباعة القيمة الحالية الموجودة في السجل A بنظام (الآسكي) ASCII ثم يعود إلى برنامجنا من جديد.

السطر الخامس من البرنامج :

يقـوم هذا السطـر بزيادة السجل A ، لأن الجولة الأولى كانت طباعة الحرف A على الشـاشـة ومـا نريـده في هذه المرة زيادة هذه القيمة الموجودة في السجل A بمقدار واحـد «1» ، وهكـذا سوف يزداد من 41 H وهي قيمـة الحـرف A إلى 42 H والتي هي قيمة الحرف B .

السطر السادس من البرنامج :

يقوم بمقارنة قيمة السجل A ليراها قد وصلت إلى القيمة Z + 1 ، فإذا لم تصل

فإن السطر السابع يقوم بالاختبار والقفز عائداً إلى العنوان NEXT ليقوم بتنفيذ ما سبق مرة ثانيـة. ومـرة أخـرى ننوه أن القضية أسهل لو أدخل السطر السادس بالشكل 1 + «Z» ولكن عنـدمـا يجمـع سيحول (أوتوماتيكياً) إلى قيمة الحرف «Z في (آسكي) ASCII (الستة عشر) مضافاً إليه واحداً وهو يساوي (5 B hex) .

السطر السابع من البرنامج :

يقـوم هذا السطـر بالقفـز النسبي وهنـا نستطيـع أن نرى الميزة في إعطاء السطر عنـوان (اسم فقـرة) لأنـا لا نحتـاج إلى حساب عدد (البايتات) التي يجب أن يقفز بها إلى الخلف كما يقوم المجمع بعمل ذلك لنا دون عناء.

وإضـافـة إلى ذلك فإنه يمكننا إضافة عدد من الأسطر بين السطر 4 والسطر 7 دون الإكـتـراث إلى أي شيء آخـر مثـل العـدد الجديد للبايتات التي يجب أن يقفز بها إلى الخلف حيث أن المجمع يقوم بحساب العدد الجديد للقفز النسبي (أوتوماتيكياً) مراعياً أن القفز سوف لا يتجاوز الـ 126- أو الـ 129 + .

السطر الثامن من البرنامج :

ينفـذ هذا السطـر فقـد عنـدمـا يكون السجل A مساوياً لـ Z + 1 ، أي عندما تنتهي جميـع الحـروف الهجـائيـة ، عندها السطر الثامن يقوم (بشحن) السجل A برمز (الآسكي) ASCII لنهـايـة السطـر والـذي يعـود بالنقطة المضيئة (Cursor) إلى الزاوية اليسـاريـة من السطـر، والسطـر رقم 9 يقـوم باستـدعـاء (روتين) الطباعة الموجود في العنوان 00 A 2H من جديد ليقوم بطباعتها.

ملاحظة :

تعتـبر رمـوز (الآسكي) (ASSCII codes) التي تحت الـ 20 hex أحـرف تحكم لموقـع النقطـة المضيئة على الشاشة. . الخ . ويمكن استخدامها باستدعاء العنوان 00A2 كما حدث للحرف الأبجدية السابقة.

السطر العاشر من البرنامج :

يقـوم هذا السطـر (بشحن) السجـل A برمـز (الآسكي) ASCII لتغذية السطر (أي النـزول إلى السطـر التـالي من الشـاشـة) فإن ما نحتـاجـه ليس الرجوع بالنقطة المضيئة إلى يسار الشاشة بل أيضاً بالتحرك إلى الأسفل بمقدار سطر أي إلى السطر

الذي يلي، وكذلك الإستدعاء يتم للعنوان 00 A2 في السطر 11 ليصار إلى تنفيذ العمل.

السطر الثاني عشر من البرنامج:

يضعنا مرة ثانية تحت سيطرة المجمع ZEN عندما ينتهي البرنامج من التنفيذ بالقفز إلى العنوان LOOP (A00 3 H). المهمة الثانية هي التأكد من أننا أدخلنا أسطر البرنامج بشكل صحيح، بالحقيقة يوجد لدينا بعض الأخطاء الواضحة التي يمكن ملاحظتها الآن، كما أن الوقوع في الأخطاء عند الإدخال في هذه المرحلة المبكرة جيد فإن اكتشافها والتعرف على العدد الشائع منها يجعلنا ذوي خبرة في المرات القادمة.

ـ أدخل «A» مع الضغط على المفتاح «RETURN»، فهذا يخبر المجمع ZEN أننا نرغب في تجميع البرنامج الذي أدخلناه قبل قليل.

الشاشة ستعطينا إرشاداً للاختيار وهو الذي يحدد إذا كنا نرغب أن نجمع البرنامج ونعطي النتيجة على الطابعة وذلك بإدخال «P»، أو إدخال «E» ليتم التجميع على وحدة خارجية (محيطية)، أو بإدخال «V» يتم طباعة النسخة المجمعة الشاشة، أو إذا ضغطنا فقط على مفتاح الـ «RETURN» فهذا يعني أن التجميع سيتم داخلياً فقط ويتم الوقوف عند السطر الذي يحوي خطأ، وهذه الطريقة هي أسرع الاختيارات. إذاً بعد ظهور إرشاد الاختيار اضغط على المفتاح «RETURN»

الشاشة سوف تظهر لك ما يلي: ORG!

2 START: CALL 0849 h

ZEN >

وبشكل بسيط يعني أننا لم ندخل مصدر البرنامج وهو المكان الذي نريد أن يتواجد به البرنامج. وفي الحقيقة يكون هذا أكبر إهمال لأننا لم نحدد مكان البرنامج وهذا شيء أساسي ومهم بالنسبة للمجمع فلا بد له من أن يعرف أين وضعنا برنامجنا في الذاكرة.

أدخل الآن «T» ثم اضغط على المفتاح «RETURN» سوف ترى السطر الأول من البرنامج قد ظهر لديك على الشاشة لأن «T» تعني السطر المطلوب إظهاره على الشاشة، أما إذا أدخلت «T4» سوف يظهر السطر الرابع على الشاشة، حيث أن عندما تدخل فقط الحرف «T» فهذا يعني أنك تريد السطر الأول فقط.

أدخل «E»، وكما قمنا أول مرة بإدخال البرنامج، سوف نقوم بإدخال أسطر جديدة على البرنامج السابق ابتداء من السطر الحالي الموجود على الشاشة، وهو السطر رقم «1» الذي ظهر على الشاشة بعد إدخال «T»، وبعد إدخال هذه الأسطر الإضافية على البرنامج فإن جميع أرقام الأسطر السابقة ستتغير وتزحف بمقدار سطر واحد، فالسطر الموجود رقم 1 سيبقى على حاله ولكنه أصبح الآن السطر الثاني بعد الإزاحة. وهكذا.

كذلك يلزم إدخال سطر بيحدد أين نرغب أن (نشحن) البرنامج في الذاكرة بعدما يكون قد جُمع وانتهى، وليس من الضروري أن يكون هذا العنوان هو عنوان المصدر نفسه (ORG) ولكن حتى نبقي هذا البرنامج بسيط غير معقد فسوف نقوم (بشحنة) بعد التجميع في المكان نفسه.

برنامج

DISPLAYED	TO ENTER
ZEN >	E
1	ORG 0E000H
2	LOAD 0E000H
3	
ZEN >	

ـ لاحظ النقطة في السطر الثالث فهي مهمة لتعيدنا مرة ثانية إلى مستوى الأمر.

أدخل الآن «T» مع «RETURN» سوف يظهر السطر رقم واحد «1»:

1 ORG 0E 000H

ZEN >

أدخل الآن «P 16» مع «RETURN» سوف تظهر على الشاشة قائمة البرنامج من السطر رقم 1 إلى نهاية البرنامج مع ظهور «EOF» وهي تعني نهاية الملف.

أما إذا أدخلنا «P8» فقط فإن أول ثمانية أسطر من البرنامج سوف تظهر على الشاشة وهكذا فإذا أردت طباعة كامل أسطر البرنامج على الشاشة فعليك إدخال

«P» متبوعة برقم آخر سطر من البرنامج أو أكبر منه . سوف تلاحظ في هذه الحالة أن الأسطر الأصلية في الذاكرة قد زحزحت بمقدار سطرين .

مرة أخرى أدخل «A» مع «RETURN» سوف يظهر لك دليل الإختيار من جديد. فضغط «RETURN» لكي نتأكد أن برنامجنا صحيح أو غير صحيح وهل سوف يجمع . فإذا أدخلنا البرنامج كما سبق فسوف يتوقف مرة ثانية وتظهر على الشاشة :

```
HuH?
6 NEXT: CALL 00 2AH
ZEN >
```

لنرى ما الخطأ في هذا السطر، ولكن الدليل «HuH?» لا يخبرنا ما نوع الخطأ بل يدلنا على أن هناك خطأ. سيظهر لك هذا الدليل كثيراً عندما تقوم بإدخال برامجك في المستقبل .

لننظر إلى هذا السطر، إنه يبدو بحالة جيدة أي لا يوجد فيه أي خطأ. ولكن في الحقيقة الخطأ في هذا السطر هو خطأ في قاعدة أساسية نسيناها لأننا لم نترك فراغاً بين الأمر CALL والعنوان 00 2 AH .

أدخل «N» مع «RETURN» سوف يظهر لك السطر والنقطة المضيئة تكون على يمينه :

```
6 NEXT: CALL 00 A2H
```

وبشكل بسيط احذف الأحرف من اليمين مستخدماً مفتاح المسافة الخلفية «BS» لأن مفاتيح النقطة المضيئة لا تعمل مع المجمع ZEN ، حتى تصل النقطة المضيئة فوق الصفر الأول بعد الأمر CALL عندها أدخل فراغ ثم 00 A2H مع «RETURN» .

السطر سوف يصبح على الشكل التالي :

```
6 NEXT: CALL  00 A2H
```

أدخل الآن «A» مع الضغط على المفتاح «RETURN» مرتين ليعطيك الحاسب النتيجة هذه المرة رسالة لا خطأ فيها، بعدها سوف يظهر دليل المعالج

«ZEN» في الحال على السطر التالي، كي يخبرنا أن التجميع هذه المرة قد تم (.O.K) وأنه قد (شحن) في الذاكرة .

أدخل «GE 00H» مع «RETURN» والشاشة سوف تظهر :-

```
BKPT >
```

هذا الدليل يسألنا أن ندخل النقطة التي نرغب أن يتوقف بها البرنامج ، لأنه إذا قمنا باختبار جزء معين من برنامج طويل فإنه يمكن إيقافه بعنوان محدد في الذاكرة والتحكم سوف يعود مرة ثانية إلى ZEN .

هذه العملية مفيدة جداً لأن برامج لغة الآلة تنفذ بشكل سريع جداً مما يجعل متابعتها صعبة للغاية .

وفي الحالة التي لا نريد فيها إدخال نقطة توقف في البرنامج فما علينا إلا الإجابة على BKPT > بالضغط على مفتاح RETURN.

سوف تمحى الشاشة ويظهر عليها ما يلي :

برنامج

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZEN>
```

لا تتوقع أكثر من ذلك من برنامج لغة الآلة هذا، لأنه كان عرضاً للمبادىء في إدخال الرموز.

لندخل الآن «A» ونرى ما قد حدث للبرنامج بعد تجميعه على الشاشة، في هذه المرة عندما ترى دليل الاختيار أدخل «V» مع «RETURN» والنتيجة سوف تكون كما يلي :

برنامج

```
PAGE    1

                        ORG   0E000H
                        LOAD  0E000H
                 LOOP:  EQU   0A003H
E000  CD4908            CALL  0849H
E003  3E41              LD    A,"A"
E005  CDA200     NEXT:  CALL  00A2H
E008  3C                INC   A
```

```
E009 FE5B        CP    "Z"+1
E00B 20F8        JR    NZ,NEXT
E00D 3E0D        LD    A,0DH
E00F CDA200      CALL  00A2H
E012 3E0A        LD    A,0AH
E014 CDA200      CALL  00A2H
E017 C303A0      JP    LOOP
                 END
ZEN>
```

في البرنامج السابق نتيجة لكونه سهلاً وبسيطاً لم نقم بتوثيق الوظائف التي يقوم بها كل سطر من أسطر البرنامج ولكن في البرامج الطويلة والمعقدة يكون من الضروري جداً وصف كل جزء بالضبط من البرنامج.

يمكن للتعليق أن يضاف إلى أي سطر وذلك بوضع فاصلة منقوطة «;» ثم تكتب بعدها التعليق المناسب لهذا السطر. فمثلاً لإضافة تعليق إلى السطر الثالث أدخل «T3» مع «RETURN» . بعدها أدخل «N» مع «RETURN» بعدها سوف يظهر السطر الثالث على الشاشة والنقطة المضيئة على يمين السطر:

3 LOOP: EQU 0A 000H

بعد ذلك قم بالإضافة التالية إلى هذا السطر:

; JUMP ON END

مع ضغط المفتاح «RETURN» طبعاً.

هذا السطر عندما يطبع من جديد سوف نرى الآن التعليق بعد الفاصلة المنقوطة والذي يذكر أي شخص في المستقبل يحاول قراءة هذا البرنامج ما مهمة هذا السطر بالضبط.

وهنا ليس كما في لغة (البيزيك) نستطيع تتمة سطر لم ينته معنا بالسطر الذي يليه حيث أن المجمع ZEN لا يسمح لنا إلا بكتابة سطر شاشة واحد، فإذا أردنا تتمة التعليق فعلينا الكتابة على سطر جديد وبدون إدخال أي أوامر فقط ندخل «;» فاصلة منقوطة ونتابع الكتابة. تستخدم هذه الطريقة في التعليقات الملحقة الكبيرة لتكون أكثر وضوحاً للقارىء.

فإذا كان لدى أحدنا طابعة يستطيع طلب قائمة البرنامج المجمع بـ «P» حيث يستطيع رؤية التعليق مطبوعاً على يمين الورقة بعد صيغة الأوامر. أما إذا طلبنا قائمة

البرنامج على الشاشة بـ «V» فإننا لا نرى التعليق يظهر على شاشة الحاسب لكون أعمدة الشاشة هي ٣٧ عموداً فقط، ما لم نكون قد أدخلنا التعليق في أسطر منفصلة ضمن البرنامج.

التعديل والإضافة:

Alterations and Additions

إن وجد من تابع وفهم البرنامج السابق وكانت جميع الأوامر السابقة واضحة بالنسبة له فإنه يستطيع تعديل البرنامج ليقوم بطباعة الأحرف الأبجدية بشكل معكوس من الحرف Z إلى الحرف A

عدل السطر رقم 5 ليصبح «Z», 5 LD A

السطر رقم 7 ليصبح 7 DEC A

السطر رقم 8 ليصبح 1- «A» 8 CP

هذا التعديل سيقوم بشحن السجل A بالحرف «Z» وبدلاً من زيادة السجل A نقوم بتنقيص قيمته بالسطر رقم 7، وهكذا فالدورة الأولى سوف تعطينا قيمة السجل A والتي هي الحرف «Y» وهكذا إلى النهاية.

في السطر رقم 8 تتم عملية المقارنة هل وصلت قيمة السجل A إلى A-1 فإذا لم تصل عاد البرنامج من جديد لطباعة بقية الأحرف وهكذا.

رسائل الشاشة:

SCREEN MESSAGES

غالباً ما نحتاج في برامجنا طباعة بعض الرسائل التي تفيد في إدخال معلومات من الشاشة إلى الذاكرة، وبما أن هذا البرنامج قصير فإن تعديله بسيط جداً، فالسطر الأول بعد تنظيف الشاشة هو سطر الاستدعاء رقم 5، أدخل «T5» مع «RETURN» فتجد السطر رقم 5 قد ظهر على الشاشة:

```
5 LD A, «Z»
ZEN >
```

أدخـــل «E» مع «RETURN» سيكـون بإمكـانـك الآن إضـافـة أسطـر على البرنامج وزحزحة الأسطر الموجودة في الذاكرة.

برنامج

DISPLAYED	TO ENTER
5	LD HL,MESG1
6	CALL 6678H
7	.
ZEN >	

هذه الأوامر الجديدة تعني ما يلي:

LD HL, MESSG1 أي (اشحن) السجـــل الزوجي HL بعنـوان في الـذاكـرة لبداية رسالة الشاشة والتي تحمل العنوان MESG1 الذي خصص لها.

CALL 6678 H هذا الأمـر يعني إستـدعـاء لروتـين في الروم ROM والذي يقوم بطباعة الرسالة التي تبدأ من العنوان الموجود في السجل الزوجي HL في الموقع الموجودة فيه النقطة المضيئة على الشاشة.

الرسالة كما سوف ترى في الأسفل من هذه الصفحة أيضاً تحوي على أي حرف من حروف التـوجيـه لتحريك النقطة المضيئة والتي يمكن أن تكون قد أدخلت قبل أو بعد القـوسـين الذين يحويان الرسالة، إضافة إلى ذلك الرسالة يجب أن تحدد برمز الـ (NOP) ,«0» (والذي يستخدم عادة كعلامة لنهاية الرسالة).

المهمـة التـاليـة هي ادخـال MESG 1 في برنامجنا. اطبع البرنامج على الشاشة حتى تعرف رقم السطر الأخير وتحدد رقم السطر الذي سوف نضع فيه رسالتنا.

طبعـاً END يجب أن تظهـر كما هو الحـال في السطر 17 لذلك أدخل «T 17» مع «RETURN» ثم أدخل «E» مع «RETURN».

برنامج

DISPLAYED	TO ENTER
17	MESG1:DB"TEST",0DH,0AH,0
18	.
ZEN>	

إذا كانت رسـالتـك أكثـر من سطر واحد عندئذٍ أنهي السطر الأول من الرسالة بإضافة قوسين علويين لإغلاق الرسالة وتابع الرسالة على السطر الذي يلي والتأكد من أنك بدأت السطر الجديد بـ «DB» وأدخل فقط «0,» في نهاية الرسالة.

بالطبع يجب إعادة تجميع البرنامج من جديد بعد إضافة الرسالة والتأكد من أنه لا يوجد شيء قد حذف خطأً من البرنامج بعد إجراء التعديل.

عنـدمـا يطبـع الـبرنـامج المجمع على الشاشة سوف لا تطبع الرسالة بالكامل على يمـين الشـاشـة ومهـما يكن (فـالبـايتـات) التي تمثل تلك الرسالة قد أدخلت في الـذاكـرة كما شوهـد على يسـار الشـاشة. أما إذا كان سطر الـ MESG1 قد أدخل كما شاهدنا سابقاً فإن الطباعة عادة سوف تبتر بعد الفاصلة المتبوعة بـ «0 DH».

لتنفيذ البرنامج يمكن ادخال «GE000H». سوف تلاحظ أن الشاشة نظفت والـرسـالـة «TEST» قد طبعت في السطـر الأول والحـروف الهجائية طبعت بعد ذلك بشكل معكوس على السطر الذي يلي.

يمكن إضـافـة رمـوز مثـل «0AH» لتكون الطباعة في أسفل الشاشة أو حسب الطلب. تأكد م أن برنامجك قد أدخل حسب ما هو موجود في القائمة لأننا سوف نقوم بتعديله بعد ذلك:

برنامج

```
 1 ORG 0E000H
 2 LOAD 0E000H
 3 LOOP:EQU 0A000H;JUMP ON END
 4 CALL 0849H
 5 LD HL,MESG1
 6 CALL 6678H
 7 LD A,"Z"
 8 NEXT:CALL 00A2H
 9 DEC A
10 CP "A"-1
11 JR NZ,NEXT
12 LD A,0DH
13 CALL 00A2H
14 LD A,0AH
15 CALL 00A2H
16 JP LOOP
17 MESG1:DB"TEST",0DH,0AH,0
18 END
EOF
```

إدخالات المستخدم -1-

USER INPUTS 1

لنفرض أننا نرغب من المستخدم أن يدخل رقم من 1 إلى الرقم 9 حتى نتمكن من طباعة الأحرف الهجائية عدة مرات.

(الروتين) موجود طبعاً في منطقة (الروم) الذي يقوم بإيقاف البرنامج وينتظر حتى يضغط المفتاح قبل المتابعة ويمكن الإستفادة من هذه الروتين بشكل سهل وبسيط تماماً.

عدل السطر 17 بإدخال «T 17» مع «RETURN» ثم أدخل «N» مع «RETURN» حتى تستطيع تعديل الرسالة، وبواسطة النقطة المضيئة من يمين السطر احذف بمفتاح المسافة الخلفي الرسالة السابقة وعدل السطر حتى يصبح على الشكل التالي:

17 MESG 1 : DB «INPUT 1 To 9», 0AH, 0 DH,0

نحتاج أيضاً في هذه الحالة إلى تغيير البرنامج حتى يقبل الإدخال عن طريق لوحة المفاتيح الأرقام من 1 إلى 9. أدخل «T7» مع «RETURN» ثم أدخل «E» مع «RETURN».

برنامج

```
                TO
DISPLAYED       ENTER

   7            TIMES:CALL 009FH
   8            CP 31H
   9            JR C,TIMES
  10            CP 3AH
  11            JR NC,TIMES
  12            SUB 30H
  13            LD B,A
  14            .
ZEN>
```

العنوان (اسم الفقرة) يجب أن يضاف إلى السطر الحالي رقم 14 لأنه من الضروري للحلقة التكرارية العودة مرة ثانية. عدل السطر 14 حتى يصبح:

14 START: LD A, «Z»

السطر رقم 7 المعنون بـ (TIMES) يقوم باستدعاء (روتين) من منطقة (الروم) (00 9 FH) الذي يوقف البرنامج وينتظر ضغط المفتاح. بينما ينضغط المفتاح يعود (الروتين) الفرعي إلى برنامجنا مع قيمة الأسكي ASCII للمفتاح المخزن في السجل A

لأننا نحتاج إلى المفاتيح من (1 إلى 9) فيجب مراقبة محتويات السجل A، فالسطر رقم 8 يقوم بمراقبة المفتاح الذي ضغط هل هو يساوي أو أكبر من 31 H والذي هو رمز (الآسكي) للرقم 1. من السهل طرح 31 H من محتويات السجل A فإذا كانت النتيجة هي عبارة عن رمز (آسكي) ASCII أقل من 31 H عندها علم الباقي سوف يوضع فيه واحد «1» (SET) ومن ثم السطر رقم 9 يقوم بالقفز النسبي عائداً إلى السطر رقم 7 من أجل إبقاء المعالج (الميكروي) في حالة انتظار حتى يتم ضغط مفتاح آخر.

فيما بعد البرنامج يجب أن يراقب القيمة العليا للمفتاح أي القيمة الأكبر من (9).

السطر رقم 10 يقارن المفتاح المضغوط مع القيمة 3 AH والتي هي عبارة عن رمز لعلامة التوقف « : » والتي تأتي بالترتيب بعد الرقم «9» في جدول (الآسكي) ASCII

السطر 11 يقوم بالقفز النسبي عائداً إلى السطر 7 إذا كانت النتيجة بعد طرح 3AH من محتويات السجل A جعلت عَلَم الباقي لا يساوي الواحد (NOT SET) وهذا يعني أن المفتاح الذي ضغط كان يساوي أو أكبر من 3 AH والذي يعني أيضاً أن المفتاح كان أكبر في جدول (الآسكي) ASCII من الرقم 9 ويجب علينا القفز إلى الوراء والانتظار لضغط مفتاح آخر.

بفرض أن المفتاح الصحيح كان قد أدخل فنحن الآن نعلم أن محتويات السجل A تحوي رقم بين الـ 31 H و 39 H ويجب علينا تحويل هذه القيمة إلى رقم بين 1 وإلى 9 والسطر رقم 12 يقوم بهذه المهمة بالضبط وذلك بطرح 30 H من محتويات السجل A ليعطي قيمة من 1 إلى 9.

السطر رقم 13 يقوم (بشحن) السجل B بمحتويات السجل A حيث السجل B هو عبارة عن عداد لعدد مرات الحروف الأبجدية التي سوف تطبع.

هناك سطر إضافي يجب ادخاله، أدخل «T 23» مع «RETURN» ثم أدخل «E» مع «RETURN».

برنامج

DISPLAYED	TO ENTER
23	DJNZ START
24	

```
ZEN>
```

هذا الأمر كان قد نوقش في قسم (القفز الخاص) في الفصل الثاني والذي هو الأمر الوحيد من أوامر المعالج Z 80 مع السجل B والذي يقوم بإنقاص السجل B وتنفيذ القفز النسبي إلى الخلف إلى أي مكان قد حددته له في البرنامج ليقوم بتنفيذ الأوامر التي في الحلقة التكرارية مرة ثانية حتى تصل قيمة السجل B إلى الصفر «0» وهذا يشبه إلى حد ما الأمر FOR/ NEXT في لغة (البيزيك) . في هذه الحالة سوف يتم القفز إلى الوراء للسطر 14 والذي قد عنون بالعنوان START .

طبعاً يجب الآن تجميع البرنامج من جديد قبل البدء بتنفيذه مرة ثانية . فإذا تكررت الأخطاء خلال عملية التجميع فارجع إلى السطر الذي فيه الخطأ وحاول إيجاد التعديل الصحيح كما فعلنا في مرحلة سابقة من هذا الفصل .

لتنفيذ البرنامج أدخل «GE 000H» مع «RETURN» أما بالنسبة لنقطة التوقف (BKPT) أدخل «RETURN» .

قائمة البرنامج بعد التجميع :

The assembled listing:-

```
PAGE    1

  1                                ORG  0E000H
  2                                LOAD 0E000H
  3                    LOOP:       EQU  0A000H          ;JUMP ON END
  4 E000 CD4908                    CALL 0849H
  5 E003 2130E0                    LD   HL,MESG1
  6 E006 CD7866                    CALL 6678H
  7 E009 CD9F00        TIMES:      CALL 009FH
  8 E00C FE31                      CP   31H
  9 E00E 38F9                      JR   C,TIMES
 10 E010 FE3A                      CP   3AH
 11 E012 30F5                      JR   NC,TIMES
 12 E014 D630                      SUB  30H
```



```
13 E016 47                        LD   B,A
14 E017 3E5A     START:           LD   A,"Z"
15 E019 CDA200   NEXT:            CALL 00A2H
16 E01C 3D                        DEC  A
17 E01D FE40                      CP   "A"-1
18 E01F 20F8                      JR   NZ,NEXT
19 E021 3E0D                      LD   A,0DH
20 E023 CDA200                    CALL 00A2H
21 E026 3E0A                      LD   A,0AH
22 E028 CDA200                    CALL 00A2H
23 E02B 10EA                      DJNZ START
24 E02D C300A0                    JP   LOOP
25 E030 494E5055 MESG1:           DB   "INPUT 1to9",0AH,0DH,0
25 E034 54203174
25 E038 6F390A0D
25 E03C 00
26   .                            END
```

إدخالات المستخدم -2- :
USER INPUTS -2-

هذا القسم يتعامل مع إدخالات المستخدم ويكون طولها غير محدد أو موصوف ، كإدخال مفتاح ابجدي رقمي بشكل تلقائي ، أو إدخال رسالة ما من لوحة المفاتيح لتطبع بعد ذلك عدة مرات .

في هذا المثال جميع العناوين قد أعطيت اسماء عناوين خاصة بها (اسم فقرة) وهي الطريقة المثلى عندما نقوم بكتابة برنامج طويل وإدخاله بشكل صحيح وجيد سوف يكون تدريب عملي حتى نتمكن من الحصول على نسخة صحيحة وبدون أخطاء .

أدخل «K» مع RETURN حتى تقوم بإلغاء البرنامج الموجود في الذاكرة وبعد ذلك أدخل «E» مع «RETURN» لتبدأ بإدخال البرنامج التالي :

برنامج

DISPLAYED	TO ENTER
1	ORG 0E000H
2	LOAD 0E000H
3	LOOP:EQU 0A003H
4	;ROM ROUTINES
5	PTMESG:EQU 6678H
6	PINLIN:EQU 00AEH

```
7     CLS:EQU 0849H
8     INPBUF:EQU 0F55EH
9     CHGET:EQU 009FH
10    CHPUT:EQU 00A2H
11    ;CONTROL CODES
12    BL:EQU 7
13    CR:EQU 0DH
14    NEWLNE:EQU 0AH
15    ;
16    CALL CLS
17    LD HL,MSG1
18    CALL PTMESG
19    CALL BELL
20    CALL PINLIN
21    CALL CRLF
22    LD HL,MSG2
23    CALL BELL

24    CALL PTMESG
25    TIMES:CALL CHGET
26    CP 31H
27    JR C,TIMES
28    CP 3AH
29    JR NC,TIMES
30    SUB 30H
31    LD B,A
32    AGAIN:CALL CRLF
33    LD HL,INPBUF
34    NEXTCH:LD A,(HL)
35    CP 0
36    JR Z,FINI
37    CALL OUTPUT
38    INC HL
39    JR NEXTCH
40    FINI:DJNZ AGAIN
41    CALL CRLF
42    JP LOOP
43    ;
44    ;OUTPUT ROUTINES
```



```
45    BELL:LD A,BL
46    JR OUTPUT
47    CRLF:LD A,NEWLNE
48    CALL OUTPUT
49    LD A,CR
50    OUTPUT:CALL CHPUT
51    RET
52    ;
53    ;MESSAGES
54    MSG1:DB"ENTER A "
55    DB"STRING",0DH,0AH,0
56    MSG2:DB"INPUT 1to9",0DH,0AH,0
57    END
58
ZEN
```

السطر رقم 16 وهو بداية البرنامج حيث يقوم باستدعاء (روتين) موجود في منطقة (الروم) ROM تحت العنوان 0849 H لتنظيف الشاشة وقد أشير إليه بالعنوان (CLS) .

الرسالة التي تفيد في إدخال صفوف أبجدية رقمية (STRING) قد (شحنت) في السجل الزوجي HL وطبعت بوساطة (الروتين) الذي تم استدعاؤه من العنوان 6678 H والموجود تحت اسم العنوان (PTMESG) .

السطر 19 يقوم بإستدعاء العنوان المدعو BELL والذي أدخل في (روتين) الإخراج في السطر 45 حيث السجل A قد شحن بالحرف المطلوب، في هذه الحالة BL (7) ، بعد ذلك يقفز البرنامج إلى الإخراج (السطر 50) حيث يتم استدعاء اسم العنوان CH PUT والتي تعبر عن العنوان (00 A2H) الموجود في الروم ROM لإخراج محتويات السجل A بعد الرجوع إلى الوراء حيث وجد السطر 20 .

هذا السطر يستدعي (الروتين) من (الروم) ROM الموجود تحت العنوان (00AEH) والمشار اليه في البرنامج بـ (PINLIN) والذي يسمح بالإدخال من لوحة المفاتيح إلى أن يتم ضغط المفتاح «RETURN» ويقوم بتخزين الصفوف الأبجدية الرقمية التي أدخلت (أي شيء تم إدخاله من لوحة المفاتيح) في منطقة الإدخال (INPUT Buffer) والتي هي تبدأ من العنوان (0F55H) والمشار إليها في البرنامج بـ (INPBUF) .

السطـر 21 يستـدعي (الـروتـين) الفـرعي الخـاص بتغذية الخط ونهاية السطر الموجـود في السطـر 47 ومـرة أخـرى السّجـل A (يشحن) بقيمة (الآسكي) ASCII لحرف التحكم أولاً من خلال العنوان (0AH) والمشار إليه في البرنامج بـ (NEWLNE) ثم يتم الإخراج عن طريق استدعاء العنوان (OUTPUT) الموجود في السطر 50 من البرنامج.

السطر 51 يعود بنا إلى السطر الذي يلي آخر إستدعاء CALL والذي هو السطر (49) حيث يتم شحن السجل A بالقيمة (0DH) الموجودة في البرنامج تحت العنوان CR وهذه المرة ينفذ البرنامج في السطر 50 لإخراج الحرف الذي في السجل A مرة أخرى.

في هذه المرة يعـود بنا السطر 51 إلى السطر الذي يلي سطر الاستدعاء الرئيسي في البـرنـامـج المـوجـود في السطـر (22) عند ذلك الرسالة الثانية (تشحن) في السجل الـزوجي HL ثم يتبـع ذلـك السطر 23 الذي يقوم باستدعاء العنوان BELL الموجود في البـرنـامـج ليقـوم (بشحن) BL إلى السجـل A ثم تطبع بعد ذلك الرسالة الثانية بعد تنفيذ السطر 24 من البرنامج.

الأسطر من 25 إلى 31 تقوم بإدخال الأرقام من 1 إلى 9 كما في البرنامج السابق.

السطـر 33 (يشحن) بدايـة منطقـة الإدخـال حيث تكـون صفـوف الأحـرف الأبحدية الرقمية مخزنة (أي شيء قد تم إدخاله من لوحة المفاتيح) في السجل الزوجي HL والسطر 34 (يشهحن) أول حرف من هذه الصفوف الأبجدية الرقمية في السجل A ثم يتم طبعها في السطر 37 الذي يقوم بإستدعاء روتين الاخراج (OUTPUT) .

عندما يخزن صف الحروف الأبجدي الرقمي في منطقة الإدخال فإن (البايت) الـذي يكـون بعد آخر (بايت) من صف الحروف الأبجدي الرقمي (تشحن) بالقيمة صفـر «0» لذلـك في السطـر 35 نقوم بمقارنة محتويات السجل A إذا كانت صفراً فإذا كانت نتيجـة الإختبـار ايجـابية فإن القفز النسبي إلى العنوان (FINI) الموجود في السطر 40 ينفذ من خلال السطر 36 .

السطر 38 يقوم بزيادة السجل الزوجي HL ليقوم بتحريكه إلى الحرف التالي في منطقـة الإدخـال والسطـر 39 يقـوم بالقفـز عائداً إلى العنوان NEXT CH في السطر 34 (ليشحن) الحرف الجديد في السجل A مرة ثانية ومقارنته هل هو صفراً أم لا .

السطـر 40 يقـوم بإنقـاص السجـل B والذي كان قد خصص كعداد، والحلقة التكـراريـة تعـود إلى السطـر 32 المعنـون بـ (AGAIN) لطباعة صف الحرف الأبجدي الرقمي مرة أخرى.

السطـر 41 يقـوم بتنفيـذ مرة أخـرى نهايـة السطـر وخـط التغـذية قبل أن يقفز الـبرنـامـج عائـداً إلى العنـوان LOOP (0A003H) والـذي هو عنـوان البداية الساخنة للمجمع NEZ .

إن الاختلاف الرئيسي بين عنواني المجمع ZEN (0A000) والعنوان (0A003) هو أن اتمام القفز إلى 0A003 سوف يحافظ على شرط السجلات ويسمح بإدخال «X» مع «RETURN» لفحص سجـلات المستخـدم . وهـذا يكـون مفيـد جداً عنـدمـا تكون البرامج كبيرة وذات أهمية .

قائمـة البرنامج المجمع التالية تبين استخدام الإختيار «P» مع «RETURN» من أجل الطابعة حيث أن الاختلاف الرئيسي بين هذا الاختيار «P» وبين «V» أي على الشاشة هو أن أرقـام الأسطـر قد شُملت في الإخراج وإن حقول التعليق قد ظهرت بشكل كامل مع الأعمدة الإضافية لها .

حتى تقـوم بتنفيـذ الـبرنامج أدخل GE00H مع «RETURN» مرتين حتى تقفز عن الـدليـل «BKPIT» ، وبعـد ذلك سوف تمحى الشاشة وتظهر الرسالة « ENTER ASTRING» . بعـد إدخـال صف الأحرف الأبجدية الرقمية سوف تظهر رسالة جديدة على الشـاشـة «INPUT 1 To 9» والتي تريـد منك أن تدخل قيمة بين (1 والـ 9) وبعد إدخال القيمة سوف يطبع صف الحروف الأبجدي الرقمي التي تم إدخالها .

```
PAGE    1.

 1                        ORG   0E000H
 2                        LOAD  0E000H
 3            LOOP:       EQU   0A003H
 4            ;ROM ROUTINES
 5            PTMESG:     EQU   6678H
 6            PINLIN:     EQU   00AEH
 7            CLS:        EQU   0849H
 8            INPBUF:     EQU   0F55EH
 9            CHGET:      EQU   009FH
10            CHPUT:      EQU   00A2H
11            ;CONTROL CODES
12            BL:         EQU   7
13            CR:         EQU   0DH
14            NEWLNE:     EQU   0AH
```

```
15
16 E000 CD4908                  CALL CLS
17 E003 2151E0                  LD   HL,MSG1
18 E006 CD7866                  CALL PTMESG
19 E009 CD42E0                  CALL BELL
20 E00C CDAE00                  CALL PINLIN
21 E00F CD46E0                  CALL CRLF
22 E012 2162E0                  LD   HL,MSG2
23 E015 CD42E0                  CALL BELL
24 E018 CD7866                  CALL PTMESG
25 E01B CD9F00    TIMES:        CALL CHGET
26 E01E FE31                    CP   31H
27 E020 38F9                    JR   C,TIMES
28 E022 FE3A                    CP   3AH
29 E024 30F5                    JR   NC,TIMES

30 E026 D630                    SUB  30H
31 E028 47                      LD   B,A
32 E029 CD46E0    AGAIN:        CALL CRLF
33 E02C 215EF5                  LD   HL,INPBUF
34 E02F 7E        NEXTCH:       LD   A,(HL)
35 E030 FE00                    CP   0
36 E032 2806                    JR   Z,FINI
37 E034 CD4DE0                  CALL OUTPUT
38 E037 23                      INC  HL
39 E038 18F5                    JR   NEXTCH
40 E03A 10ED      FINI:         DJNZ AGAIN
41 E03C CD46E0                  CALL CRLF
42 E03F C303A0                  JP   LOOP
43                ;
44                ;OUTPUT ROUTINES
45 E042 3E07      BELL:         LD   A,BL
46 E044 1807                    JR   OUTPUT
47 E046 3E0A      CRLF:         LD   A,NEWLNE
48 E048 CD4DE0                  CALL OUTPUT
49 E04B 3E0D                    LD   A,CR
50 E04D CDA200    OUTPUT:       CALL CHPUT
51 E050 C9                      RET
52                ;
53                ;MESSAGES
54 E051 454E5445  MSG1:         DB   "ENTER A "
54 E055 52204120                DB   "STRING",0DH,0AH,0
55 E059 53545249
55 E05D 4E470D0A
55 E061 00
56 E062 494E5055  MSG2:         DB   "INPUT 1to9",0DH,0AH,0
56 E066 54203174
56 E06A 6F390D0A
56 E06E 00
57                              END
```

تخزين البرامج :
SAVING PROGRAMS

مما نحتاجه أيضاً في بعض الأحيان تخزين البرامج التي نكتبها على شريط

مغناطيسي ، فنحاول تخزين هذا البرنامج الذي لدينا وكونه للتدريب حتى نستوعب عملية التخزين بشكل صحيح وتام .

إن عملية التخزين هنا ليست سهلة أو قريبة من عملية تخزين برنامج كتب بلغة (البيزيك) ، وهكذا فالوقوع في الخطأ الآن أفضل والضرر سوف يكون أقل مما لو حدث هذا الخطأ في المستقبل لبرنامج قد كتبته بلغة الآلة وأخذ منك الجهد الكبير حتى أنجز.

في المجمع ZEN طريقتين في تخزين البرامج التي كتبت بلغة الآلة :

- الطريقة الأولى هي تخزين الملف الأساسي على شكل ملف نص (بالآسكي) ASCII أي البرنامج يخزن (بالآسكي) ASCII .

الملفات بنص (الآسكي) أو برامج مخزنة (بالآسكي) ASCII تكون مؤلفة من نص صافٍ كما أدخل من لوحة المفاتيح دون أي تغيير أو تبديل عليها .

ربما أحدنا يحتاج هذه الطريقة في تخزين برنامج لم ينته بعد وبشكل أوضح حيث يمكن تجميعه على وضعه الحالي ، ويمكن شحن هذا البرنامج في المستقبل ستخدام المجمع ZEN الذي يقوم بانجاز هذه المهمة بادخال «R» مع «RETURN» بعد ظهور دليل ZEN .

أدخل «H» مع «RETURN» سوف تظهر لديك الآن على الشاشة بداية ونهاية الملف الأساسي وأعلى منطقة الذاكرة . في هذه المرحلة البرنامج الأخير سوف يظهر :

C000 C2A7 F37F

إذا لم يقم أحد بإضافة فراغات أو تعليقات جديدة .

فإذا أدخلنا QC000H4 مع «RETURN» فإن النص الذي أدخل سوف يشاهد في الذاكرة (بايت) بعد (بايت) .

لحفظ برنامج (بالآسكي) ASCII بإستخدام ZEN أدخل «W» مع «RETURN» بعدها يلقن اسم الملف ويكون البرنامج قد خزن على الشريط بشكل عادي .

يجب بعد ذلك التحقق والتأكد من تخزين الملف على الشريط .

- الطريقة الثانية لتخزين ملف بلغة الآلة (Object) كملف بالنظام الثنائي (Binary) (الملف بالنظام الثنائي هو عبارة عن برنامج مجمع) .

وفي الحقيقة ما نقوم بتخزينه هو عبارة عن ملف مخزن برموز الآلة فقط دون أي تعليقات وجاهز للتنفيذ . في برنامجنا الأخير يمكن تخزينه ، ثم بعد ذلك تنفيذه مباشرة بعد شحنه من غير وجود المجمع ZEN ، بوساطة الأمر السيط BLOAD نحتاج أيضاً إلى تعديل السطر 42 من البرنامج من الأمر JP LOOP إلى الأمر RET حيث لا نحتاج في هذه المرحلة القفز إلى العنوان A003 لأن المجمع ZEN يكون غير مشحون .

عدل الآن السطر 42 حتى يصبح على النحو التالي :

42 RET

طبعاً نحتاج هذه المرة أيضاً إلى تجميع البرنامج من جديد بعد تعديله ولكن إذا كان التعديل البسيط السابق صحيحاً يمكننا هذه المرة التجميع على الشاشة وذلك إدخال «V» مع «RETURN» كما يجب أن نعرف عنوان النهاية للملف .

بعد تعديل السطر 42 سوف يكون طول البرنامج أقصر (ببايتين) من النسخة السابقة قبل التعديل وهذا ما يجعل نهاية البرنامج عند العنوان E06CH .

ضع الآن شريط (كاسيت) جديد وأدخل «WB» والتي تعني الكتابة بالنظام الثنائي . سوف ندخل طبعاً عنوان البادية والذي هو «E000H» مع «RETURN» ومن الضروري أيضاً إدخال اللاحقة «H» وإلا سوف يعتقد المجمع ZEN أن هذا الرقم هو رقم عشري ولكنه غير ذلك .

بعد ظهور الدليل التالي سوف يكون الإدخال لعنوان التنفيذ (EXEC) لأنه العنوان الذي يبدأ البرنامج التنفيذ منه . في هذه الحالة نحن نريد لهذا البرنامج أن ينفذ من نفس العنوان الذي (شحن) منه ولهذا أدخل مرة أخرى «E000H» مع «RETURN» . في الحقيقة يضاف عنوان التنفيذ EXEC لأن البرنامج ليس دائماً يبدأ التنفيذ من عنوان البداية في الذاكرة . حيث يمكن لذلك البرنامج أن يكتب وبعد ذلك أضيف له بعض رسومات العناوين على الشاشة في نهايته ولكن من الذي نريده أن ينفذ أولاً فعنوان التنفيذ يمكن أن يكون مختلفاً عن عنوان (الشحن) .

بعد ذلك سوف تظهر رسالة تدل على دليل (الشحن) LOAD لإدخال العنوان الذي يجب شحن البرنامج فيه ومرة أخرى أدخل «E000H» .

الرسالة الأخيرة التي تظهر من أجل إدخال اسم البرنامج الذي نريد تخزينه

فيمكننا وبشكل بسيط أن نختار لهذا البرنامج الاسم «TEST» لأنه برنامج اختبار وبعد ذلك لا يبقى علينا إلا تجهيز المسجلة بوضعية النسخ «RECORD» .

بمجرد أن ينتهي البرنامج من التخزين أطفىء جهاز (الكومبيوتر) ثم انتظر عدة ثوان (لا تقم بإطفاء الكومبيوتر ثم اشعاله بسرعة) أعد تشغيل (الكومبيوتر) (واشحن) البرنامج الاختباري الذي قمنا بتخزينه وذلك بإدخال :

BLOAD «TEST», R

بعد ثوانٍ سوف يقوم البرنامج بالعمل بشكل (أوتوماتيكي) إذا كنت قد قمت بتخزينه بشكل صحيح وبعد أن ينتهي من التنفيذ سوف يقفز إلى حلقة لغة (البيزيك) BASIC الأساسية وتظهر الرسالة «OK» .

* نرجو أن يكون واضحاً لك أن هذا عبارة عن تمرين فقط لعملية التخزين بشكل صحيح وبعد ذلك (الشحن) والتنفيذ لبرنامج مكتوب بلغة الآلة ومن الطبيعي أن يكون أكثر إثارة من كونه برنامج اختبار فقط ، فقد أخذ منا جهداً ليس كبيراً من ساعات البرمجة حتى أصبح صحيحاً في هذه المرحلة فلو كتب هذا البرنامج بشكل خاطىء فإننا سوف نضيع عدة ساعات من العمل لتعديله .

السقوط :
CRASHES .

عندما نقوم باختبار البرامج في المجمع ZEN فما الذي يمكن اجراؤه عند وجود خطأ ما في برنامجك أدى إلى سقوط هذا البرنامج من سيطرة المجمع ZEN وهذا ما يدعى بـ (CRASHES) وبعد خروجه عن تحكم ZEN فيمكن أن يعود إلى (البيزيك) BASIC أو حتى يمكن أن يعاد العمل من جديد وكأن (الكومبيوتر) قد أشعل الآن فتظهر رسالة MSX على الشاشة معلنة عن بداية جديدة .

يمكننا العودة إلى المجمع ZEN بإدخال :

DEF USR = & HA000

A = USR (0)

بعد ذلك نرجو أن يكون المجمع ZEN وبرنامجك قد بقيا في الذاكرة حيث يستطيع المتابعة بعد اكتشاف الخطأ.

وهذا أيضاً يمكن ان يحدث عند معالجة (روتين) (بيزيك) من برنامج مكتوب بلغـة الآلـة حتى إذا تكـرر الخطأ فإن (روتـين) مصيـدة الخطأ المـوجـود في (البيزيك) باستطـاعتـه أن يلتقـط هذا الخطأ ويظهر رسالة الخطأ ويفرغ محتويات الذاكرة ويعطي بعد ذلك رسالة الـ «OK» والتي تعني أننا في طور (البيزيك).

وبجعـل إعـادة التخـزين بسيطـاً وسهلاً يمكننا إدخال سطرين في الأعلى بلغة (البيـزيـك) معطيـاً لكـل سطـر رقماً بالطبع فإذا لم يقطع السقوط أيضاً اضغط المفتاح «F5» والذي يعني «RUN» لإستعادة السيطرة للمجمع ZEN من جديد.

الفصل الرابع
روتينات الـ MSX
MSX Routines

هذا الفصل يتعرض لعـدد من (الـروتينـات) التي زودت بها (روم) ROM الـ MSX وطريقة تداولها.

بناء الجدول
TABLE Construction

الـبرنـامج التالي يستخدم لوحة إدخال المفاتيح لإصدار نغمات في المجال من C إلى B في أي طبقة موسيقية من الطبقات الثمانية (8-OCtaves) التي تعطيها جاذباً عند السـماع، ولكن الغـرض الأسـاسي من كل هذا هوعرض طريقـة واحـدة من طرق معالجة الجداول.

المفاتيح التي سوف تصدر الأصوات هي كما يلي:

RT UIO

D F G H J K I

السطـر السفـلي يستخـدم للنغمات من C إلى B بينما المفاتيح في السطر العلوي تشير إلى حدة النغمة (C + . . . إلخ). المفتاح «£» يستخدم للخروج من البرنامج.

عنـد بدايـة تنفيذ البرنامج تكون الطبقة الموسيقية (4) (4 Octave) ولكن يمكن تعديلها حسب الطلب وذلك بضغط المفاتيح من 1 إلى 8 خلال عمل البرنامج.

النغمـة الحـاليـة والطبقـة المـوسيقيـة يظهران على الشاشة والبرنامج يستخدم

الـروتـين PLAY للمـوسيقى المـوجـود في لغـة (البيـزيك) والموجود في العنوان 73E5H ولـذلـك صف الحـروف الـذي سوف يعطي النغمات الموسيقية يجب أن يحاط بقوسين علويين (« ») تماماً كما تستخدم (الروتين) PLAY في (البيزيك) (PLAY «CB») .
ويجب أن ينتهي (بـــايت) قيمتهـــا صفــراً وإلا سوف يقــع خطأ وســوف يعــود البرنامج إلى طور (البيزيك) مع إظهار رسلة الخطأ على الشاشة .

يمكننا الآن بعد أن أصبحنا متمرنين في ادخال البرامج إدخال القائمة المجمعة التي لدينا مع الإشارة إلى أنه سوف يتم شرح جميع روتينات الروم ROM المساعدة التي استخدمت في هذا البرنامج .

```
1                            ORG   0E000H
2                            LOAD  0E000H
3                   QUIT:    EQU   0A003H       ;ZEN MAINLOOP
4                   CHGET:   EQU   009FH        ;WAIT FOR KEY
5                   CLS:     EQU   00C3H        ;CLEAR SCREEN
6                   POSIT:   EQU   00C6H        ;CURSOR SET UP
7                   PTMESG:  EQU   6678H        ;PRINT MESSAGE
8                   CLIKSW:  EQU   0F3DBH       ;KEY CLICK SW
9                   CHPUT:   EQU   00A2H        ;OUTPUT CHARACT.
10                  ERAFNK:  EQU   00CCH        ;ERASE FUNC KEY
11                  CHSNS:   EQU   009CH        ;KEY SCAN
12                  ;
13 E000 CDCC00      START:   CALL  ERAFNK       ;FUNC KEYS OFF
14 E003 AF                   XOR   A            ;ZERO A
15 E004 32DBF3               LD    (CLIKSW),A   ;TURN OFF CLICK
16 E007 CDC300               CALL  CLS          ;CLEAR SCREEN
17 E00A 2608                 LD    H,8          ;SET CURSOR COLUMN
18 E00C 2E02                 LD    L,2          ;SET CURSOR LINE
19 E00E CDC600               CALL  POSIT        ;POSITION CURSOR
20 E011 21CCE0               LD    HL,MESG1
21 E014 CD7866               CALL  PTMESG
22 E017 260C                 LD    H,12
23 E019 2E0A                 LD    L,10
24 E01B CDC600               CALL  POSIT
25 E01E 21E0E0               LD    HL,MESG2
26 E021 CD7866               CALL  PTMESG
27 E024 260E                 LD    H,14
```



```
28 E026 2E0C                 LD    L,12
29 E028 CDC600               CALL  POSIT
30 E02B 21E9E0               LD    HL,MESG3
31 E02E CD7866               CALL  PTMESG
32 E031 C34CE0               JP    PTOCT            ;PRINT OCTAVE VALUE
33 E034 CD9C00      INPUT:   CALL  CHSNS            ;IS KEY DOWN
34 E037 28FB                 JR    Z,INPUT          ;NO LOOP BACK
35 E039 CD9F00               CALL  CHGET            ;GET KEY IN REG A
36 E03C FE23                 CP    "£"              ;IS IT £ KEY
37 E03E CA03A0               JP    Z,QUIT           ;YES FINISH
38 E041 FE31                 CP    31H              ;TEST FOR 1
39 E043 38EF                 JR    C,INPUT          ;IF LESS GET NEXT
40 E045 FE39                 CP    39H              ;TEST FOR 9
41 E047 3012                 JR    NC,SAMOCT        ;STILL SAME OCTAVE
42 E049 3295E0               LD    (OCTVE+1),A      ;DISPLAY OCTAVE
43 E[illegible] 2615          PTOCT:   LD    H,21             ;POSITION CURSOR
44 E04E 2E[illegible]A               LD    L,10             ;TO PRINT OCTAVE No.
45 E050 CDC600               CALL  POSIT
46 E053 3A95E0               LD    A,(OCTVE+1)      ;NEW OCTAVE
47 E056 CDA200               CALL  CHPUT            ;PRINT IT
48 E059 18D9                 JR    INPUT            ;GET NEXT KEY
49 E05B CD9000      SAMOCT:  CALL  0090H            ;NO QUEUES
50 E05E 47                   LD    B,A              ;SAVE KEY IN B
51 E05F 21A7E0               LD    HL,TABLE
52 E062 7E          COMPR:   LD    A,(HL)           ;TABLE IN A
53 E063 FE0F                 CP    0FH              ;END OF TABLE?
54 E065 28CD                 JR    Z,INPUT          ;YES WRONG KEY
55 E067 23                   INC   HL
56 E068 B8                   CP    B                ;COMPARE KEY/TABLE
57 E069 2804                 JR    Z,FOUND          ;GO PLAY
58 E06B 23                   INC   HL               ;NOT FOUND. BUMP OVER
59 E06C 23                   INC   HL               ;NOTE STRING AND
60 E06D 18F3                 JR    COMPR            ;TEST NEXT IN TABLE
61                  ;
62 E06F 7E          FOUND:   LD    A,(HL)           ;NOTE TO PLAY
63 E070 32A3E0               LD    (NOTE),A
64 E073 23                   INC   HL
65 E074 7E                   LD    A,(HL)           ;SECOND PART OF NOTE
66 E075 32A4E0               LD    (NOTE+1),A
```

```
67 E078 2193E0              LD   HL,STRING        ;HL=PLAY STRING
68 E07B CDE573              CALL 73E5H            ;BASIC PLAY ROUTINE
69 E07E 2615                LD   H,21             ;POSITION CURSOR TO
70 E080 2E0C                LD   L,12             ;RIGHT OF NOTE:-
71 E082 CDC600              CALL POSIT
72 E085 3AA3E0              LD   A,(NOTE)         ;PRINT CURRENT
73 E088 CDA200              CALL CHPUT            ;NOTE, AND
74 E08B 3AA4E0              LD   A,(NOTE+1)       ;PRINT + CHARACTER
75 E08E CDA200              CALL CHPUT             ;OR SPACE
76 E091 18A1                JR   INPUT             ;GET NEXT KEY
77                 ;
78 E093 22         STRING:  DB   22H               ;START WITH QUOTES
79 E094 4F34       OCTVE:   DB   "O4"              ;OCTAVE 4
80 E096 543630     TEMPO:   DB   "T60"             ;TEMPO 60
81 E099 4C38       DURAT:   DB   "L8"              ;DURATION 8
82 E09B 5330       ENVPAT:  DB   "S0"              ;ENV WAVEFORM S0
83 E09D 4D313030   ENVPER:  DB   "M10000"          ;PERIOD M10000
83 E0A1 3030
84                 NOTE:    DS   2                 ;NOTE STORAGE
85 E0A5 22                  DB   22H               ;PLAY END QUOTES
86 E0A6 00                  DB   0                 ;END STRING WITH 0
87                 ;
88 E0A7 444320     TABLE:   DB   "D","C "
89 E0AA 524323              DB   "R","C+"
90 E0AD 464420              DB   "F","D "
91 E0B0 544423              DB   "T","D+"
92 E0B3 474520              DB   "G","E "
93 E0B6 484620              DB   "H","F "
94 E0B9 554623              DB   "U","F+"
95 E0BC 4A4720              DB   "J","G "
96 E0BF 494723              DB   "I","G+"
97 E0C2 4B4120              DB   "K","A "
98 E0C5 4F4123              DB   "O","A+"
99 E0C8 4C4220              DB   "L","B "
100 E0CB 0F                 DB   0FH                ;END OF TABLE MARKER
101                 ;
102 E0CC 43555252 MESG1:    DB   "CURRENT NOTE "
102 E0D0 454E5420
102 E0D4 4E4F5445
```



```
102 E0D8 20
103 E0D9 53544154           DB   "STATUS",0
103 E0DD 555300
104 E0E0 4F435441 MESG2:    DB   "OCTAVE:-",0
104 E0E4 56453A2D
104 E0E8 00
105 E0E9 4E4F5445 MESG3:    DB   "NOTE:-",0
105 E0ED 3A2D00
106                         END
```

من ناحيـة عمليـة فإن الأسـماء أو العنـاوين (اسم فقـرة) تشير إلى (روتينات) (الـسروم) ROM التي استخـدمت طبقـاً لمواصفـات نظـام الـ MSX وينبغي أن تكـون متـوافقـة مع النشـرات الأخـرى للـ MSX ، حيث لديهم على الأكثر ستة أحرف هي عبـارة عن اختصـار للوظـائف التي تقـوم بها هذه (الروتينات) ـ الإختصار (CHGET) يشير إلى (الروتين) الذي يقوم بتلقي الحرف من لوحة المفاتيح (CHaracter GET)

تحليل البرنامج :
ANALYSIS

السطر : 13-(CALL ERAFNK (00CCH

يلغي وظائف المفاتيح من F1 إلى F10 التي تظهر على الشاشة .

أما (الروتين) المعاكس الذي يعيد وظائف هذه المفاتيح إلى الشاشة فهو:

CALL DSPENK (00CFH)

السطر 15- (CLINKSW (F3DBH

يلغي تكـة المفتـاح عند الضغط (CLICK) لأنه يقوم بتصغير السجل A وذلك بـ (XOR A) ويشحن داخـل F3DB بالقيمـة صفـر «0» الـذي يقوم لإلغاء (التكة) وأي قيمة أخرى يمكن أن تعيد هذه (التكة) من جديد .

السطر 16 - (CALL CLS (00C3H

ينظف الشـاشـة ولكن فقط السجل A قد نظف بالأمر (XOR A) 00C3H تشمل القفز إلى (روتين) تنظيف الشاشة الحقيقي في العنوان (0848H) .

إذا كنت ترغب في تنظيف الشاشة ولكنك غير متاكد من محتويات السجل A فإن الـ -CALL 0849H- سوف تقوم بهذه المهمة وذلك بتجاوز الاختبار على العلم . لقد استخدمت هذه الطريقة في الفصل السابق .

السطر 19 - CALL POSIT (00C6H)

يقوم بوضع النقطة المضيئة على الشاشة معتمداً على قيمة السجل HL في السطر 17 والسطر 18 حيث العمود ادخل في «H» والسطر ادخل في «L» .

السطر 21 -CALL PTMESG (6678H)

يطبع الرسالة من عنوان البداية الموجود في السجل HL ويجب أن تنتهي هذه الرسالة بصفر. MESG1 يمكن أن نراها في السطر 102 .

الأسطر 22-31 تتكرر العملية كما في الأسطر السابقة حيث تمت طباعة الرسالة الثانية والثالثة على الشاشة بسطرين مختلفين وذلك بشحن «H» و «L» بقيم عشرية ولم تستخدم قيم ستة عشر (hex) .

السطر 27-28 : يمكن أن يدخلا بسطر واحد وذلك بتحويل القيم العشرية إلى (ستة عشر) حيث تصبح 14 = 0E و 12 = 0C

ويختصر السطران إلى سطر واحد حسب ما يلي :

LD HL, 0E0CH ما يجعل البرنامج أقصر.

السطر 33- :

CALL CHSNS (009CH)

يقوم بفحص مخزن لوحة المفاتيح ، مكان تخزين المفتاح المضغوط ، فإذا ضُغط مفتاح ما يقوم بتصفير العَلَم Z (Reset) . أما السطر 34 يقوم بالقفز النسبي عائداً إلى السطر 33 إذا لم يكن هناك مفتاح مضغوط حيث يقوم بالانتظار لضغط أي مفتاح . البرنامج لن يتخطى هذين السطرين حتى يتم إدخال أي مفتاح .

السطر 35 - CALL CHGET (009FH)

ينتظر حتى يتم ضغط مفتاح ما ليعود بقيمة (الأسكي) ASCII له ويضعها في السجل A . في الحقيقة باستطاعتنا الاستغناء عن السطرين 33/34 بوجود هذا (الروتين) الذي يقوم بانتظار ضغط أي مفتاح ولكن أيضاً تظهر النقطة المضيئة على

الشاشة لتدل على الانتظار، ومن وجهة نظري الشخصية (وهذا للمؤلف) هذا يقوم بتخريب الإظهار على الشاشة ، ولذلك استخدام (الروتين) CHSNS هو أول وسيلة للتأكد أن البرنامج لن يصل إلى هنا حتى يكون المفتاح في المخزن الوسيط وبعد ذلك يقوم هذا (الروتين) بإلتقاط المفتاح ولا يحتاج إلى أي انتظار ولهذا فإن النقطة المضيئة لن تظهر على الشاشة .

السطر 36 -Checks-

يقوم بالمراقبة إذا ضغط المفتاح «£» . فإذا تم ذلك فالسطر 37 ينهي البرنامج بالخروج. هذا السطر يقوم بالقفز عائداً إلى المجمع ZEN ولكن إذا قمنا بتخزين هذا البرنامج على شكل ملف بالنظام الثنائي «Bainary» ونفذناه بدون المجمع ZEN فهذه التعليمة ينبغي أن تعدل إلى RET Z .

الأسطر 38/41 ـ يقوم بمراقبة المفاتيح من 1 إلى 8 لتغير اللحن طبقاً لما ورد في المراقبة السابقة للمفاتيح أما إذا كان المفتاح المضغوط أكبر من 8 فإن البرنامج يقفز إلى العنوان SAMOCT في السطر 49 للتحقق من القيمة مع النغمة الموسيقية للعزف .

السطر 42 ـ يتم تنفيذ هذا السطر إذا كان المفتاح المضغوط بين 1 و 8 و (يشحن) القيمة في الـ OCTAVE + 1 في المكان الذي خزن به اللحن .

الأسطر 43/45 ـ يقوم بوضع النقطة المضيئة على الشاشة بقرب رسالة اللحن وذلك بإستدعاء العنوان POSIT.

السطر 47 - CALL CHPUT (00A2H)

يقوم بطباعة الحرف الذي في السجل A الذي كان قد (شحن) في السطر 46 في **موضع النقطة المضيئة الحالية التي حددت الآن في الأسطر 43/45 والسطر 48 يقوم** بالقفز عائداً إلى إدخال المفتاح التالي .

باستطاعتنا تعديل السطر 47 من CALL CHPUT إلى RST 18H الذي يقوم باخراج السجل A إلى وحدة محيطية ، طابعة ، شاشة ، أي شيء آخر .

السطر 49 - CALL 0090H (GICINI)

يقوم باختبار البدء لمولد الصوت المبرمج (PSG) وقد استخدم لحذف صف النغمات الموسيقية المخزنة والتي لا تقوم بمتابعة العزف من عدة دقائق بعد ضغط المفتاح . يمكنك حذف هذا السطر للحصول على نتائج مختلفة .

السطر LD HL, TABLE - 51

(يشحن) السجل HL بعنوان البداية لجدول النغمات الموسيقي في السطر 88 . المفتاح الـذي خزن في السجـل B في السطر 50 وأول إدخال للجدول (قد شحن) في السجل A في السطر 52 . السطر 53 يقارن السجل A مع القيمة 0FH والتي تشير إلى نهاية الجدول فإذا لم يوجد المفتاح فهذا يعني أن مفتاح غريب وغير موجود في الجدول مثلاً المفتاح X أو Z إذا أدخل فانه لن يحدث شيء ولن يكون أي عزف موسيقي وسوف يتم القفز للوراء لانتظار إدخال مفتاح جديد.

السطـر 55 يقـوم بزيـادة السجـل HL بمقـدار (بايت) واحد والسطر 56 يقارن المفتـاح المضغـوط في السجل B مع الجدول في السجل A فإذا كان هناك تطابق فسيتم القفز النسبي إلى FOUND لعزف النغمة الموسيقية (الجولة الأولى ستكون المقارنة مع أول مفتاح مخزن في الجدول والذي هو المفتاح D) .

إذا لم يتم التطـابق عنـدهـا السجـل HL يجب أن ينتقل إلى البايت التالية والتي يجب أن تشـير الآن إلى المفتـاح «R» في السطـر 89 ، تذكـر أنه قد ازداد السجل HL مرة قبل ذلك وفي السطرين 58/59 يزداد السجل HL مرتين أياً والسطر 60 يقوم بالقفز إلى الوراء ليقارن الإدخال التالي في الجدول.

هذه المقارنة تستمر حتى يصل السجل HL إلى (البايت) الأول في السطر 100 وهي (0FH) عبـارة عن علامـة نهايـة الجـدول ولهـذا فقد كان السطران 53/54 يقومان بمراقبة نهاية الجدول للعودة ثانية إلى إدخال جديد.

السطر FOUND - 62

يصـل البرنـامـج إلى هذا السطر عندما يتم تطابق المفتاح المضغوط مع مفتاح الجدول عندها يجب أن يتم العزف الموسيقي . تذكر أن السجل HL يشير إلى الجدول وقد تمت زيادته في مرحلة سابقة (السطر 55) .

لينفرض أن المفتاح «D» قد أدخل والسجل HL سوف يشير الآن إلى (البايت) الذي بعد «D» في السطر 88 من الجدول . هذا الحرف يكون من النغمة الموسيقية التي تعزف ولـذلـك قد شحن في السجـل A (السطر 62) . السطر 63 يشحن هذه النغمة داخل الموقع في صف الحروف والمعنون بـ NOTE في السطر 84 .

في الحقيقة لقد حجزت بايتين لهذه النغمة الموسيقية في السطر 84 بإدخالها على الشكل «DS 2» . الأول هو حرف النغمة بينها (البايت) الثاني استخدم لتخزن إشارة الموجب « + » .

النغمة الأولى في الجدول (السطر 88) هي نغمة متبوعة بـ « + » ولذلك أدخلنا فراغ بعـد النغمـة ـ الفـراغ مسمـوح به بين صف الحـروف في تعليمـة PLAY في (البيـزيـك) فهو لا يدل على شيء ولن يؤدي أي عمل ـ الذي سوف يؤدي فيها بعد للكتابة فوق النغمة الموسيقية التي خزنت في السابق مع إشارة الموجب « + » إذا كانت موجودة.

(لشحن) القسـم الثاني من النغمة الموسيقية السطر 64 يقوم بزيادة السجل HL والسجـل A يشحن أيضـاً إما بفـراغ أو بالإشـارة «+» والسطـر 66 يقـوم بتخزينها بداخل NOTE + 1 والتي هي عبارة عن البايت الثانية من المخزن.

السطر CALL 73E5H - 68

يستـدعي (روتـين البيـزيـك) من أجـل تعليمة العزف PLAY والتي تحتاج إلى بدايـة صف الحـروف لتقـوم بالعزف والتي تكون في السجل HL حيث السطر 67 ينفذ هذه المهمة

الأسطر 69/71 : ـ يقـوم بوضع النقطة المضيئة طبقاً للنغمة الموسيقية التي تعزف حيث تظهر الرسالة على الشاشة متطابقة مع النغمة الموسيقية التي تنفذ.

الأسطر 72/75 : ـ يشحن النغمـة المـوسيقيـة في السجـل A ويطبعها بالأمر CALL CHPUT لاستخدامها في طباعة اللحن في السطرين 43/47 . لأن النغمة دوماً مؤلفة من حرفين السجل A يكون التالي مشحوناً (بالبايت) التالي من مخزن النغمة الموسيقية وبطريقة مشابهة يقوم بالإظهار في السطر 75 .

النقطـة المضيئـة لا تحتـاج إلى تجهيـز موقع لها على الشاشة من أجل (البايت) التالي فإنها سوف تنتقل (أوتوماتيكياً) بمقدار موقع واحد على الشاشة بعد تنفيذ سابق للـ CHPUT في السطـر 73 (طبـاعـة سابقـة) . بعـد ذلك يتم القفز من جديد لإدخال مفتاح تالي (JP INPUT) .

السطر STRING- 78

المكان الذي خزن فيه صف الحروف كاملاً التي يستخدمها الأمر «PLAY» .

السطر 78 يحوي قيمة (الأسكي) ASCII للقوسين العلويين '«' والتي يجب أن تفتح عند بداية صف حروف المعزوفة وتغلق عند نهايته "....." .

كما نحن نعالج أوامر لغة (البيزيك) بشكل صحيح تماماً حتى لا يظهر خطأ يؤدي إلى إيقاف البرنامج كذلك أيضاً عند الخطأ هنا سوف يؤدي إلى اسقاط برنامج لغة الآلة والعودة ثانية إلى طور (البيزيك) .

الأسطر 79 يخزن اللحن مع الإستهلال بالحرف «O» ، (ليس صفراً) ويتبع بقيمة البدء «4» .

في الحقيقة البايت التالي يبدل إذا ضغط أحد المفاتيح من 1 إلى 8 عندما يكون البرنامج في طور التنفيذ مما يؤدي إلى تغير اللحن .

الأسطر 80/83
يحدد سرعة عزف الموسيقا لصف الحروف بـ (T60) خلال مدة (L8) وكذلك يحدد غلاف شكل الموجة بـ (S0) وفترة الغلاف بـ (M10000) .

هذه القيم تكون ثابتة ولا يمكن تعديل إلا النغمة واللحن من خلال لوحة المفاتيح . بالطبع يمكن تعديل هذه الثوابت حسب الطلب وبعد ذلك يعاد التجميع للبرنامج من جديد حيث لا يتطلب أكثر من ثواني للقيام بذلك والحصول على نتائج مختلفة . يمكن تعديل البرنامج ليقبل مفاتيح النقطة المضيئة لاجراء تعديل على سرعة العزف أو المدة أو شكل الموجة .

السطر 84ـ يحوي هذا السطر مخزن النغمة الموسيقية الذي يكون فراغ عندما ينفذ البرنامج لأول مرة .

السطر 85 هو عبارة عن قيمة (الأسكي) ASCII للقوس العلوي '«' والسطر 86 يحوي (بايت) قيمته «0» صفراً والتي يجب أن تدخل لتشير إلى نهاية صف الحروف .

الأسطر 88/100ـ يحوي جدول المفاتيح متبوعة بالنغمات الخاصة بها والسطر 100 يحوي علامة نهاية الجدول والتي هي (0FH) .

الأسطر 102/105ـ هو عبارة عن الرسائل التي تطبع وتكون متناسبة مع الحالة الآن بحيث تكون متبوعة (ببايت) قيمته صفر بعد كل واحد منها .

لحفظ ملف على شكل ترميز ثنائي استخدم الأجراءات نفسها في الفصل السابق .

عدل السطر 37 إلى RET Z ولاحظ (البايت) عندما يظهر لك على الشاشة رسالة تريد منك عنوان التوقف (STOP address) .

هذا البرنامج قد كتب لينفذ على الشاشة ذات (الموديل) «0» ولكن من الممكن أيضاً أن ينفذ على الشاشة (موديل) «1» ما عدا أن الإظهارات سوف تكون منحرفة قليلاً إلى اليمين لأن هذا لن يزعج (روتين) الـ (POSIT) الذي يحدد موضع النقطة المضيئة على الشاشة .

بإستطاعتنا إدخال سطر في البداية ليجهز نوع الشاشة حسب ما يلي :

برنامج

```
CALL 006FH
(INIT32) will initialise to screen 1.

CALL 006CH
(INITXT) initialises screen 0.
```

الخطافات :

HOOKS

لقد خُصص في نظام الـ MSX ذاكرة تبدأ من الموقع FD9A إلى FFC9 والتي تعرف بمنطقة الخطاف (HOOK AREA) .

هناك حوالي 112 خطاف كل واحد منها يتألف من خمسة (بايتات) (5 BYTES)

(روتينات) متعددة داخل منطقة الروم ROM تقوم باستدعاء هذه الخطافات (HOOKS) لترى إذا كانت هذه الخطافات تحوي أوامر إضافية للمهات التي يجب تنفيذها .

عادة كل هذه الخطافات تحوي القيمة (C9hex) وهو عبارة عن رمز لأمر الرجوع «RETURN»

بشكل بسيط تماماً (الروتين) في منطقة (الروم) يستدعي الخطاف (HOOK) ليجد أنه يجب عليه العودة وعدم القيام بأي شيء والرجوع منه إلى المكان الذي أقلع منه .

بعد أن أصبحت أنظمة البرمجة المتطورة متاحة لنظام الـ MSX فهذه الخطافات (HOOKS) سوف تستخدم لربط وحدة (الديسك) (الاسطوانة المغناطيسية) والوحدات المحيطية الأخرى من أجل توسيع الأنظمة من دون الحاجة إلى تغيير الـ ROM .

من أجل الكتابة إلى خطاف (HOOK) يجب علينا بوضوح أن نعرف من أي (روتين) موجود في (الروم) يكون استدعاءه، وهكذا فإن الاستخدام المشوش قد يسبب الوقوع في الإشكالات، فكما يقولون اذا لم تكن متأكداً من القاعدة فدعها.

البرنامج التالي يقوم بكتابة أوامر بنموذج واحد للخطاف (HOOK) عند العنوان FD9F والمعنون في البرنامج بـ (HTIMI) ، حيث استدعي من (روتين) توقيت موجة الإعاقة والذي يعني أنه يقوم بالمعالجة خمسين مرة في الثانية مهما تكن نوع المهمة التي ينفذها نظام الـ MSX ما عدا طبعاً القراءة أو الكتابة على شريط مغناطيسي . هذا بوضوح مما يجعله يستخدم في بناء أداة التوقيت (Timer) لأنه يمكننا أن نعلم كم مرة سوف يعد في الثانية الواحدة.

لقد استخدم في البرنامج التالي هذا الخطاف لتبطيء حركة الأشباح حيث بدون هذا التأخير سوف يتحركون بشكل سريع جداً مما يصعب على العين مشاهدتهم.

في الحقيقة يمكننا كتابة روتين بلغة الآلة للقيام بهذا التأخير ولكننا نريد القيام بعرض لكيفية استخدام هذا الخطاف (HOOK) .

الأشباح :
SPRITES

إن معالج وحدة العرض المرئية (VDP) المستخدم في الـ MSX في الحقيقة هو قوي جداً وفعال وربما للوهلة الأولى يظهر لبعض المستخدمين أنه معقد وصعب.

إن معرفتك بهذه الأمور تعتمد على كمية المعلومات التي توجد في كتاب التشغيل للآلة التي لديك.

من أجل أن تحصل على أفضل الاستخدامات لمعالج وحدة العرض المرئية

(VDP) في نظام MSX يجب على الأقل أن تكون مطلع على أوامر متغيرات نظام المعالج. لوحدة العرض (في البيزيك) وكيفية معالجة السجلات المتنوعة.

هذا لسوء الحظ لا يمكن أن يوصف في فصل واحد فهذا المجال أبعد من أن يطرق في مقدمة عن لغة الآلة.

لزيادة معرفتك بهذه الأمور وكيفية عملها ينصح بقراءة بعض الكتب المتخصصة في المعالج لوحدة العرض المرئية (VDP) لنظام MSX ككتاب بعنوان.
(Behind The Screen of the MSX) فسوف تجد فيه معظم الإجابات عن أسئلتك.

المهمة الأساسية للبرنامج التالي هي إحداث نموذج لشبحين وتحريك الأول عبر الشاشة حتى يتصادم مع الآخر عند ذلك سوف يتحرك إلى أعلى الشاشة.

هذا البرنامج فقط عبارة عن عرض لكيفية إحداث شبح وتحريكه على الشاشة وملاحظة التصادم، ولكن في الحقيقة هو أيضاً عبارة عن تدريب على الكتابة بلغة الآلة.

الشرح حول هذا البرنامج سوف يأتي بعد قائمة التجميع للبرنامج.

برنامج

```
1                      ORG   0E000H
2                      LOAD  0E000H
3             CHSNS:   EQU   009CH
4             HTIMI:   EQU   0FD9FH
5             ERAFNK:  EQU   00CCH
6             WRTVDP:  EQU   0047H
7             RDVRM:   EQU   004AH
8             WRTVRM:  EQU   004DH
9             INIT32:  EQU   006FH
10            RG1SAV:  EQU   0F3E0H
11            STATFL:  EQU   0F3E7H
12            ATTR1:   EQU   1B00H
13            ATTR2:   EQU   1B04H
14            ;
15            ;WRITE CODE TO HOOK (HTIMI)
16 E000 219EE0         LD    HL,CODE
17 E003 119FFD         LD    DE,HTIMI
18 E006 010300         LD    BC,3
```

```
19 E009 EDB0                    LDIR
20
21 E00B CDCC00                  CALL ERAFNK         ;TURN OFF FUNC KEYS
22 E00E CD6F00                  CALL INIT32         ;SCREEN 1
23                      ;
24                      ;ALTER SPRITE PATTERN
25                      ;GENERATOR BASE ADDRESS TO 0000
26 E011 AF                      XOR  A
27 E012 47                      LD   B,A
28 E013 0E06                    LD   C,6
29 E015 CD4700                  CALL WRTVDP
30                      ;
31                      ;ALTER BIT 0 OF VDP REG 1
32                      ;TO 1.  TO INCREASE MAGNITUDE
33 E018 3AE0F3                  LD   A,(RG1SAV)
34 E01B F601                    OR   1
35 E01D 47                      LD   B,A
36 E01E 0E01                    LD   C,1
37 E020 CD4700                  CALL WRTVDP
38                      ;
39                      ;SET UP SPRITE 1
40 E023 3E8C                    LD   A,140          ;VERTICAL POS
41 E025 21001B                  LD   HL,ATTR1
42 E028 CD4D00                  CALL WRTVRM
43 E02B 3EC8                    LD   A,200          ;HORIZ POS
44 E02D 21011B                  LD   HL,ATTR1+1
45 E030 CD4D00                  CALL WRTVRM
46 E033 3E41                    LD   A,65           ;CHARACTER 65=A
47 E035 21021B                  LD   HL,ATTR1+2
48 E038 CD4D00                  CALL WRTVRM
49 E03B 3E01                    LD   A,1            ;COLOUR BLACK
50 E03D 21031B                  LD   HL,ATTR1+3
51 E040 CD4D00                  CALL WRTVRM
52                      ;
53                      ;SET UP SPRITE 2
54 E043 3E8C                    LD   A,140          ;VERTICAL POS
55 E045 21041B                  LD   HL,ATTR2
56 E048 CD4D00                  CALL WRTVRM
57 E04B 3E1E                    LD   A,30           ;HORIZ POS
```



```
58 E04D 21051B                  LD   HL,ATTR2+1
59 E050 CD4D00                  CALL WRTVRM
60 E053 3E42                    LD   A,66           ;CHARACTER 66=B
61 E055 21061B                  LD   HL,ATTR2+2
62 E058 CD4D00                  CALL WRTVRM
63 E05B 3E0F                    LD   A,15           ;COLOUR WHITE
64 E05D 21071B                  LD   HL,ATTR2+3
65 E060 CD4D00                  CALL WRTVRM
66                      ;
67 E063 AF                      XOR  A
68 E064 3298E0                  LD   (COUNT),A      ;ZERO COUNTER
69                      ;DELAY COUNTER CHECK
70 E067 CD9C00          CKMOVE: CALL CHSNS
71 E06A 2024                    JR   NZ,QUIT
72 E06C 3A98E0                  LD   A,(COUNT)
73 E06F FE01                    CP   1
74 E071 38F4                    JR   C,CKMOVE       ;IF LESS DONT MOVE
75
76                      ;MOVE SPRITE 2
77 E073 21051B                  LD   HL,ATTR2+1     ;VERT POS
78 E076 CD4A00                  CALL RDVRM          ;PUT INTO A
79 E079 3C                      INC  A              ;MOVE 1 PIXEL RGHT
80 E07A CD4D00                  CALL WRTVRM         ;NEW POS OF SPRT 2
81 E07D AF                      XOR  A
82 E07E 3298E0                  LD   (COUNT),A      ;ZERO COUNTER
83                      ;CHECK FOR COLLISION
84 E081 3AE7F3                  LD   A,(STATFL)
85 E084 CB6F                    BIT  5,A            ;TEST COLLISION BI
86 E086 28DF                    JR   Z,CKMOVE       ;IF ZERO KEEP MOVIN
87                      ;
88                      ;COLLISION OCCURED
89 E088 21041B                  LD   HL,ATTR2       ;HORIZ POS
90 E08B 3E28                    LD   A,40
91 E08D CD4D00                  CALL WRTVRM         ;MOVE IT UP
92                      ;
93                      ;PROG END, SO REPLACE RET
94                      ;INTO HOOK (HTIMI)
95 E090 3EC9            QUIT:   LD   A,0C9H
96 E092 329FFD                  LD   (HTIMI),A
```

```
97 E095 C303A0                      JP    0A003H
98                   ;
99 E098 00           COUNT:         DB    0
00                   ;
01                   ;INCREMENT COUNT 50 TIMES A SEC
02 E099 2198E0       INCCNT:        LD    HL,COUNT
03 E09C 34                          INC   (HL)
04 E09D C9                          RET
05                   ;
106 E09E C399E0      CODE:          JP    INCCNT
107                  ;
108                                 END
```

تحليل البرنامج :

الأسطـر 16/19ـ (يشحن) الـرمـز داخـل الخطـاف (hook) المعنون بـ (HTIMI) عند العنـوان FD9F . الأمـر LDIR استعمـل (لشحن) الـرمـز من العنـوان المشـار إليه بواسطة السجـل HL إلى العنـوان المشـار إليـه بـ السجـل DE . القيمـة التي ستنقل موجود في السجل BC ، في هذه الحالة ثلاثة (بايتات) التي سيتم نقلها كما يظهر في السطر 106 حيث يخبر الخطاف للقفز الى INCCNT الذي يقوم بزيادة العداد . في الحقيقة هذا هو تأثير تبطيء الحركة ويمكن تسريعها أو تبطيئها كما سوف نشاهد بعد ذلك .

السطر 21ـ هو عبارة عن (روتين) لتنظيف الشاشة من وظائف المفاتيح الموجودة عليهـا ويتبـع هذا السطـح استـدعاء للعنوان INIT32 الذي يقوم بوضع الشاشة على الموديل «1» لأننا لا نستطيع إحداث أشباح على الشاشة «0» .

السطـر 26/29ـ يقـوم بالكتـابـة إلى سجـل معـالج الاظهار المرئي (VDP) لأن الأرقـام من (0 الى 7) يجب أن تكـون في السجـل C والمعلومـات التي (ستشحن) إلى داخل الـ VDP موجودة في السجل B .

هنا نقوم (بشحن) سجل VDP (6(C بـ (0(B .

هذا في الواقع يكون عبارة عن تعديل عنوان الأساس لجدول نموذج الشبح إلى عنـوان أسـاس مولـد الحـرف . هكـذا فنحن الآن لدينـا (الأسكي) ASCII لمجموعة الحرف مخزنة كنموذج شبح .

السطر 33ـ (بشحن) قيمة RG1SAV (F3E0) التي تخزن القيمة الحالية للسجل 1 لمعالج الاظهار المرئي (VDP) . (البايت) رقم «0» من الـ VDP هي التي نهتم بها حيث أنها تراقب تكبير الاشباح .

الصفـر هو الحجم الطبيعي بينـما عنـدمـا نعـدلـه إلى 1 فإن الأشباح سوف تكبر ، هكـذا في السطـر 34 - OR 1ـ الـذي سوف لن يؤ ثـر في بقية البتات ولكن سوف يحول البت «0» الى 1 حيث يضعها في وضع التكبير .

ومـدة أخـرى يجب أن نشحن محتـويـات السجـل A في داخـل السجـل B ونختـار سجل VDP من السجل C ونقوم باستدعاء WRTVDP الذي سوف يكتب إلى سجل 1 من سجلات معالج الاظهار المرئي .

الآن إذا ظهـرت هذه الأشبـاح يشكـل هزيـل فلا تخف أو تشعر بالخيبة ، فهذه الممارسـة تجعلك أفضل ، فقـط أدخـل الـرمز وعدله بعد فترة فإن هذا سيكون أسهل عليك لادراك السبب .

الأسطر 40/51ـ يقوم بإحداث الشبح 1 .

الجدول لرمز الشبح في الشاشة(موديل 1) يبدأ من العنوان 1B00H ويحوي أربع (بـايتـات) لكـل شبـح . لذلـك الرمز للشبح 2 سوف يبدأ عند العنوان 1B04H حيث أعطيا في السطر 12/13 معادلتين هما ATTR1 و ATTR2 .

(البـايت) الأول تمسك موقـع المـربـع العمودي للشبح (Pixel) ، والسطر 40 (يشحن) السجـل A بالقيمـة 140 . مربعـات (Pixel) الشـاشة تكون من الصفر «0» (من الأعلى) الى 191 (من الأسفل) .

السطـر 41ـ (يشحن) السجـل HL بعنـوان رمـز الشبح 1(1B00H) ATTR1 والسطر 42 (يشحن) السجـل A في داخـل VRAM (مـراجعة شرح لـ VRMA من البيزيك) . عند العنوان 1B00H بأمر الاستدعاء CALL WRTVRM .

تستمر المعالجة (بشحن) الموقع الأفقي في داخل السجل A ويخزن في (البايت) الثاني للرمز التابع للشبح 1 عند العنوان 1B01H (بشحن) السجل HL بـ ATTR1 + 1

لاحظ : أنه يمكننا إدخال السطر 47 على الشكل التالي : TNC HL فعندما يعود البرنامج من الروتين WRTVRM يكون السجل HL لم يتغير بعد على أي حال ولذلك

فسـوف يظـل يشـير إلى المـوقع السابق وبسهولة باستطاعتنا زيادته . ولكن لقد كتبناه بتلك الطريقة لتسهيل التوضيح والابتعاد عن التعقيد .

(البـايت) الثـالث للرمـز يَمسُـك رقم حرف الشبـح . ولكننـا لم نحـدد الشبح الخـاص بنـا فقـد قمنا بزحزحة جدول نموذج الشبح ولهذا فقد بدا على شكل أحرف (آسكي) ASCII ليكون ملائماً بشكل تام .

لذلك (فالآسكي) ASCII للحرف A هو 65 عشري والسطر 46 يشحن السجل A بالقيمـة 65 حيث أن الحـرف A يكـون هو شبحنـا الأول (Sprite 1) . بعد ذلـك يشحن في داخل البايت الثالثة من الرموز 2 + ATTRA1 .

لاحـظ أنـه أيضـاً القيمـة العشرية قد استخدمت هنا فقط للتوضيح حيث قام المجمع بعد ذلك بتحويلها إلى (الستة عشر) (hex) كما هو مشاهد في العمود اليساري من قائمة البرنامج .

أخـيراً يجب تحديـد اللون وشحنـه داخل (البايت) الرابع للرمز ATTR1+3 ، والسطـر 49 يقـوم بتحـديـد اللون رقم 1 والذي يعبر عن اللون الأسود وبهذا نكون قد أنهينا احداث الشبح 1 بتخزين الحرف واللون من العنوان 1B00 الى العنوان 1B03

الاجراءات السابقة هي نفسها لإحداث الشبح 2 ما عدا المعلومات يجب أن تخزن من العنوان 1B04H الى العنوان 1B07H والذي قد عنون بـ ATT2 في السطر 13

الموقع العمودي هو نفسه كما في الشبح رقم 1 ولكن الفرق يكون فقط في الموقع الأفقي (30) في السطر 57 والحرف يكون هذه المرة «B» بدلاً عن «A» والذي يساوي 66 عشري في (الآسكي) ASCII واللون هو عبارة عن الرقم 15 الذي يمثل اللون الأبيض .

طبعـاً بعـد تنفيـذ البـرنـامـج يمكن تغيير هذه الاحداثيات حتى تعطي نتائج مختلفة مع التنويه الى ضرورة اعادة تجميع البرنامج بعد التعديل بالطبع .

الأسطـر 68/76 ـ يقوم بتصغير السجل A و(شحنه) الى COUNT لتصغيره أيضاً .

السطر 70/71 ـ ينفذ (الروتين) CHSNS الذي يقوم بالتحقق اذا تم ضغط أي مفتـاح ليقـوم بالقفز إلى (روتـين) الخروج QUIT ولقد أضيف هذا السطر إذا مارغب أحد بإيقاف البرنامج قبل أن ينتهي لتعديله مثلاً في حال يريد أن يبطيء حركة الشبح أكثر مما كانت عليه .

السطر 72 ـ يقوم (بشحن) القيمة الموجودة في COUNT العداد الى السجل A فإذا كانت لم تصل إلى الـواحـد 1 فالسطر 74 يقوم بالقفز الى الوراء عائداً الى السطر 70 ليقوم بالتحقق من جديد . هذه الحلقة تستمر حتى تكون قيمة السجل A تساوي أو أكبر من القيمة الموجودة في السطر 73 .

تذكر الخطاف (hook) عند FD9F قد قام بزيادة العداد COUNT خمسين 50 مرة في الثانية لذلك CP1 سوف تسبب فقط باستمرارية الحلقة من أجل 1/50 من الثانية قبل أن تقوم بتنفيذ البرنامج ونقل الشبح بمقدار مربع واحد .

اذا عدل السطر 73 الى CP50 فسوف يتحرك الشبح بمقدار مربع واحد في كل ثانية مما يجعله بطيء جداً .

بدون هذا التأخير فإنـه عنـد الخطـاف HTIMI (hook) سوف يتحـرك الشبـح بسرعة كبيرة مما يجعله يصل الى نقطة النهاية بسرعة .

السطـر 77/82 ـ يقـوم بتحـريـك الشبح رقم 2 ويصفر العداد من أجل التأخير التالي قبل أن يتحرك مرة ثانية .

ما نغيره فقط بالرمز هو الموقع الأفقي 1 + ATTR2 لذلك يجب أن يتم (شحنه) في داخل السجل HL ونقوم باستدعاء القراءة لـ VRAM (RDVRM) وسيعيد القيمة التي في السجل A (الموقع الافقي الحالي) .

لتحـريـك الشبـح نقوم بزيادة السجل A ونكتبه لعنوان الـ VRAM الذي ما زال مشار إليه في السجل HL بوساطة (الروتين) WRTVRM

السطر 84/86 يقوم بمراقبة تصادم الشبحين .

STATFL (F3E7H) يقوم بضبط حالة سجل معالج الاظهار المرئي (VDP) واذا تم التصادم فالبت الخامسة يوضع فيها «1» أما الشبحان فهما يتطابقان جزئياً على الأقل في مربع واحد . لذلك السطر 85 يختبر البت الخامسة (5) لحالة العلم والسطر 86 يقوم بالقفز إذا كانت تساوي الصفر عائداً الى CK MOVE مرة أخرى ، إذا حدث التصادم

(فـالبت) الخـامسـة سوف يكون يساوي الواحد لذلك علم الصفر سوف لا يساوي الواحد والاختبار سوف يفشل وسينتقل البرنامج الى السطر التالي .

السطـر 90ـ لايتم الـوصـول الى هذا السطـر إلا بعد حدوث التصادم وسوف يقـذف بالشبـح 2 الى أعلى الشـاشـة وذلـك (بشحن) السجـل HL بالـرمز العمودي ATTR2 ويبقي الرمز الأفقي 1 + ATTR2 على حالة و(يشحن) السجل A بالقيمة 40 ويستدعي WRTVRM مرة ثانية لإعادة وضعه من جديد على الشاشة .

السطر 95ـ Quit سطر الخروج .

يتم الـوصـول الى هذا السطـر بعـد حدوث التصـادم أو بعد ضغط أي مفتاح خلال تنفيـذ الـبرنـامـج بسهـولة يتم إعادة وضع محتويات الخطاف HTIMI (بالبايت) الأصلي له وهو رمز RET ويساوي (C9H).

السطر 99ـ هو عبارة عن (بايت) مخزن للعداد .

السطـر 102/104ـ يُعَـالَجان فقط من الخطاف(hook) HTIMI ويضاف واحد 1 الى العداد كل مرة حدث فيها الانقطاع .

برنامج التحميل أو الشحن
LOADER PROGRAM

عندما نشحن برنامج لغة الآلة ربما نرى رسائله مختلفة تظهر على الشاشة تكون غير مألوفة لدينا أو يمكن أن يعدل الإظهار كلياً إلى عنوان بياني يظهر أثناء (شحن) البرنامج .

الجواب في الحقيقة يكمن عندما (يشحن) برنامجان ، الثاني (يشحن) بشكل (أوتوماتيكي) .

الأول برنامج قصير يحوي عناوين وقفز إلى (روتين) (لشحن) البرنامج الكبير الثـاني . عنـدمـا (يشحن) برنامج الأول فإنه ينفذ في الحال ويقوم بطبع العناوين على الشاشة ويُدخل (روتين الشحن) للبرنامج الثاني . التنفيذ يكون سريع جداً بعد ذلك يتـوقف الشـريـط لوقت قصـير جداً ثم يبدأ من جديد وعلى الأغلب بدون أن تلاحظ أي شيء سوف تظهر على الشاشة رسالة : «Found: Program name»

(خلال شحن البرنامج الأول ؛

برنامج التحميل يبدأ برسالة على الشاشة ، حيث تظهر الرسالة التالية :

«NOW LOADING MAIN PROGRAM»

الآن أقوم (بشحن) البرنامج الرئيسي :

ينصـح إضـافـة فقـط قسم القفـز لبرنامج التحميل بعد اتمام التفقد والاختبار للعناوين البيانية . كذلك إيضاً الـ ORG توضع عند العنوان 9000H ، الذي هو مثال للاختبـار ، قبـل تخزين الـبرنامج على شكل ملف بلغة الآلة يمكن تعديله إلى موقع آخـر من الـذاكـرة وهـذا أيضـاً يعني أن البرنامج الثاني يمكن أن يوضع عند نفس الـ ORG قبل التخزين وبرنامج التحميل سوف يكتب فوقه تماماً وسوف يختفي من الذاكرة .

إذا سجلنا ملف مرمز (بـالآسكي) ASCII لبرنامج كتب للتدريب فإنه سوف يكون من السهل علينا إختبار برنامج التحميل هذا .

أولاً : أتمم الادخـالات التي على الصفحـةالتـاليـة ، تأكد من أنها تنفذ بشكل صحيـح ، عنـدئذٍ أضف قسم برنامج التحميل بعناية وخزنه على شكل ملف بالنظام الثنائي بواسطة الأمر «WB». أعد التحقق من الشريط ولا تعد لف الشريط حيث ان البرنامج الرئيسي سوف يُسجل ابتداء من مكان أول نهاية على القسم التالي من الشريط .

نظف الـذاكـرة من برنـامج التحميل وشحن البرنامج السابق الذي تم تجميعه وخـزن الـبرنـامج الثاني باستخدام الأمر «WB» . (أي تخزين بالنظام الثنائي Write Binary) على الشـريـط . لقـد أصبح لدينا الآن برنامجان على شريط واحد الثاني منهم سوف (يشحن) وينفذ بشكل (أوتوماتيكي) .

برنامج

```
1                   ORG   9000H
2                   LOAD  9000H
3       INIT32:     EQU   006FH
4       ERAFNK:     EQU   00CCH
5       FORCLR:     EQU   0F3E9H
6       BAKCLR:     EQU   0F3EAH
```

```
7                     BDRCLR:     EQU  0F3EBH
8                     CHGCLR:     EQU  0062H
9                     T32NAM:     EQU  1800H
10                    LDIRVM:     EQU  005CH
11                    FILVRM:     EQU  0056H
12                    ;
13 9000 CDCC00                    CALL ERAFNK
14 9003 3E09                      LD   A,9
15 9005 32EAF3                    LD   (BAKCLR),A
16 9008 32EBF3                    LD   (BDRCLR),A
17 900B 3E01                      LD   A,1
18 900D 32E9F3                    LD   (FORCLR),A
19 9010 CD6F00                    CALL INIT32
20 9013 210018                    LD   HL,T32NAM
21 9016 3ED1                      LD   A,0D1H
22 9018 010003                    LD   BC,768
23 901B CD5600                    CALL FILVRM
24 901E 212D90                    LD   HL,DISPL
25 9021 116319                    LD   DE,T32NAM+355
26 9024 011A00                    LD   BC,26
27 9027 CD5C00                    CALL LDIRVM
28 902A C300A0                    JP   0A000H
29                    ;
30 902D 204E6F77 DISPL:           DB   " Now Loading"
30 9031 204C6F61
30 9035 64696E67
31 9039 204D6169                  DB   " Main Program "
31 903D 6E205072
31 9041 6F677261
31 9045 6D20
32                                END
```

تحليل البرنامج

ANALYSIS

السطر 13 : يقوم بمسح وظائف المفاتيح من على الشاشة .

السطر 14/18 : يقوم بتجهيز الألوان . (يشحن) السجل A أولاً برمز لون الأحمر الفاتح ، ثم (تشحن) محتويات السجل A إلى كل من المخزن (BAKCLR) الذي يمثل

أرضية الشاشة وإلى المخزن (BDRCLR) الذي يمثل الأطار للشاشة وذلك عند العنوانين (F3EA) , 3f3eb4 على التوالي

مخزن لون الحرف (FORCLR) (يشحن) بمحتويات السجل A حيث هذه المرة تساوي الواحد «1» والتي تمثل اللون الاسود وذلك عند العنوان (F3E9)

السطر 19 : يقوم بتحضير الشاشة ذات (الموديل) رقم 1 وذلك بتنظيفها وتغير اللون إلى الألوان الجديدة السابقة .

ملاحظة : إذا رغب أحد إلى تعديل الألوان الموجودة مع ابقاء (موديل) الشاشة على حاله والمحافظة عل ما هو مُظهر حالياً على الشاشة فإن عليه استدعاء (الروتين) (CHGCLR) عند العنوان (0062H) وذلك بعد تجهيز الألوان .

السطر 20 : يقوم (بشحن) السجل الزوجي HL ببداية جدول الاسم للشاشة ذات الموديل 1 والذي هو (1800H) ، أي الموقع اليساري العلوي من الشاشة .

السطر 21 : يقوم (بشحن) السجل A برمز الحرف الذي سوف يغطي الشاشة بينما يقوم السطر 22 بشحن عداد (البايت) (السجل BC) برقم الموقع الذي سوف نقوم بالكتابة عليه وحيث أن الشاشة موديل 1 تحوي 24 سطراً وكل سطر مؤلف من 32 حرفاً فإن عداد (البايت) (BC) (شحن) بالقيمة 768 عشري ، بالطبع سوف تحول إلى نظام (الستة عشر) لتصبح على الشكل التالي :

- LD BC , 300 H -

السطر 23 : يقوم باستدعاء (الروتين) (FILVRM) عند العنوان 0056H الذي يقوم بكتابة المعلومات التي في السجل A الى ذاكرة العرض (VRAM) حيث السجل الزوجي HL يحوي عنوان المصدر لها والسجل BC يحوي عدد (البايتات) المراد نقلها .
فإذا ما أراد أحد تعديل الطباعة مثلاً إلى السطر العاشر من الشاشة حيث يبدأ من الموقع 288 (9 * 32 باعتبار السطر الأول «0») أي هو أعلى من موقع بداية الشاشة ، لذلك إذا أردنا البداية من السطر العاشر يجب أن يعدل السطر 20 حتى يصبح : (LD HL, T32NAM + 288) ولكن لا بسد من تعديل عداد (البايت) (BC) أيضاً وذلك (بشحنه) بالقيمة العشرية 288 والتي هي عبارة عن (9 * 32).

الأسطر 24/26 يقوم (بشحن) بداية رسالتنا الاعتيادية المعنونة بـ (DISP)

«Now Loading Main Program»

«الآن يتم (شحن) البرنامج الرئيسي»

يقوم (بشحنها) إلى السجل HL والسطر 25 يقوم (بشحن) عنوان موقع المقصد لهذه الرسالة على الشاشة في السجل DE و(يشحن) طول هذه الرسالة في السجل BC (عداد البايت) والذي هو عبارة عن 26 (بايت) بما فيها الفراغات لأن الرسالة تبدأ من الموقع الثالث من الجهة اليسارية للسطر 12 فإن عملية حساب هذا الموقع تتم على الشكل التالي (3 + 11 * 32) ، السطر الأول من الأعلى يكون دوماً «0» لذلك لحساب «12» سطر تقوم بضرب طول السطر بالعدد «11» وليس بالعدد «12».

السطر 27 : يقوم باستدعاء (الروتين) LDIRVM عند العنوان 005Ch الذي إستخدم في الجزء الأول (لشحن) ذاكرة العرض (VRAM) مباشرة بمحتويات منطقة معينة من الذاكرة (RAM).

السطر 28 : يقوم بالقفز عائداً إلى طور المجمع ZEN, وسوف يعدل عنوان القفز هذا عندما نقوم بإضافة قسم (الشحن) إلى البرنامج .

بعد إدخال «G9000H» سوف يتبدل الاظهار إلى الشاشة ذات (الموديل) «1» ويغطي الشاشة رمز (الأسكي) ASCII للحرف (D1) وتظهر رسالتنا على سطر المركز (منتصف الشاشة) .

قم بتنفيذ البرنامج أولاً وبعد ذلك أجري التعديلات التي نريدها مثل اللون وطول الرسالة . . . . الخ، وتذكر أن تقوم بتجميع (Assemble) البرنامج بعد إجراء كل تعديل والا لا تدخل التعديلات الى داخل الذاكرة .

بعد إجراء كل اختبار رسالة المعالج «zen» سوف تظهر في أعلى الشاشة من أجل تنظيف الشاشة والرجوع إلى موديل الشاشة «0» «Screen 0» والتي هي الشاشة العادية في المجمع ZEN ، إضغط المفتاح RETURN بعد كل رسالة .

برنامج التحميل (الشاحن) :
The Loader

لإضافة (روتين) التحميل إجعل السطر الذي يحوي الرسالة «END» السطر

الحالي، الذي من المفروض أن يكون السطر 32 إذا لم يقم أحد بتعديل البرنامج ، وأدخل «E» مع «RETURN» وأضف الأسطر التالية بدقة :

برنامج

```
              TO
DISPLAYED     ENTER

   32         LOADER:LD A,0FEH
   33         LD (0F41CH),A
   34         LD HL,STRING
   35         JP 6EC6H
   36         STRING:DB 22H,"CAS:"
   37         DB 22H,2CH,"R",0
   38
ZEN
```

السطر 28 : والذي هو عبارة عن القفز إلى المجمع ZEN يحتاج إلى تعديل حتى يصبح : JR LOADER

هذا التعديل يجعل البرنامج يقوم بالقفز إلى (روتين) التحميل بعد ظهور العناوين على الشاشة .

تأكد من أن السطر الأخير هو «END» وجمع البرنامج على الشاشة عن طريق (V) ولاحظ (البايت) الأخير من البرنامج . فإذا لم يقم أحد بتعديل البرنامج فإنها سوف تكون 905AH ولكن يمكن أن يكون برنامجك أطول من هذا البرنامج . القسم النهائي من البرنامج سوف يجمع على الشكل التالي :

برنامج

```
32 9047 3EFE      LOADER:     LD    A,0FEH
33 9049 321CF4                LD    (0F41CH),A
34 904C 215290                LD    HL,STRING
35 904F C3C66E                JP    6EC6H
36 9052 22434153  STRING:     DB    22H,"CAS:"
36 9056 3A
37 9057 222C5200              DB    22H,2CH,"R",0
38                            END
```

وبشكل بسيط تماماً لقد شحنا القيمة (FE hex) في داخل محتويات F41H والتي توقف الرسالة «Found:» للحظة عندما (يشحن) البرنامج الثاني والتي تتلف الاظهارات على الشاشة .

السطر 34 : (يشحن) السجل HL ببداية مجموعة الحروف التي عادة تتبع الأمر BLOAD والتي هي (R ,«CAS:»).

بعد ذلك نقوم بالقفز إلى (روتين) BLOAD في ذاكرة (الروم) ROM عند العنوان (6EC6H).

إذا رغبت بتخزين البرنامج السابق على شكل ملف بالنظام الثنائي (Binary) أدخل الأوامر التالية :

برنامج

```
WB
START 9000H
STOP 905AH
LOAD 9000H
EXEC 9000H
```

أعط برنامج التحميل اسم البرنامج الرئيسي الذي سوف يتبعه على الشريط .
بعد لف الشريط وإجراء التحقق من التخزين عن طريق الأمر «VB» مع «RETURN» فإذا كان كل شيء على ما يرام حرك الشريط مقدار بسيط حتى تترك فراغ بين نهاية برنامج التحميل وبداية البرنامج الثاني .

الآن نظف الذاكرة من الملف الموجود فيها (البرنامج) وذلك بإدخال «k» مع «RETURN» ، مشابهاً لتعليمة «NEW» في لغة (البيزيك) (BASIC) ، وقم (بشحن) البرنامج الرئيسي أو أدخله من خلال لوحة المفاتيح . إذا كان أحد قد خزن البرامج السابقة في الفصل الثالث كملفات آسكي ASCII فسوف يكون ذلك كمثال للإختبار فقط . حاول الآن تعديل الـ ORG و(شحن) البرنامج الثاني إلى العنوان (9000H) ، نفس العنوان الذي استخدمه برنامج التحميل ، وجمع البرنامج على الشاشة عن طريق الأمر (V) ideo وذلك حتى تلاحظ (البايت) الأخير من البرنامج وخزن هذا

البرنامج كما مر معنا في الطريقة السابقة على نفس الشريط الذي يحوي برنامج التحميل ولكن اضغط «RETURN» عندما تظهرلك رسالة إدخال اسم البرنامج .

للاختبار إطفىء (الكومبيوتر) لعدة ثوانٍ ثم أدره من جديد ، ثم أدخل من على الشاشة BLOAD «CAS:», R

وأول برنامج يوجد على الشريط سوف يشحن وينفذ وبعد ذلك بشكل أوتوماتيكي سوف يشحن البرنامج الثاني وينفذ (البرنامج الرئيسي) .

النقل باستخدام USR

Argumert Transfer Using USR.

لقد استخدمت وظيفة (USR) في الفصل الأول من هذا الكتاب للإشارة الى الدليل في داخل الأقواص () مثال ذلك USR (0).

لأنه من الممكن التمرير (لروتين) لغة الآلة عدد صحيح أو صف حروف (String) أو متغيرات ذات دقة أحادية أو مضاعفة .

A= USR (8H1234) سوف يخزن في منطقة التخزين في الذاكرة (RAM) القيمة (1234) الستة عشرة . ويستدعي (روتين) لغة الآلة .

ان سوء استخدام هذه الوظيفة (USR) يمكن أن يؤدي إلى تخريب البرنامج .
إن نوع الدليل الذي يمكن تمريرة إلى (روتين) لغة الآلة يخزن دائماً عند العنوان (F663)، لهذا فإن الروتين يمكن اجراء الفحص على نوع الدليل الذي مر إليه والذي يتألف من :

2 من أجل العدد الصحيح (integer) .
3 من أجل صف الحروف (String) .
4 من أجل متغير ذو دقة أحادية (Single Precision) .
8 من أجل متغير ذو دقة مضاعفة (Double Precicion) .

لنأخذ المثال التالي :
أولاً نحتاج إلى تعريف (DE FINE) لعنوان المستخدم (USER) كما شاهدنا في الفصل الأول من الكتاب . (10 DEF USR 2 = & HE 000).

إن (روتين) لغة الآلة الذي لدينا يتم استدعاءه من لغة (البيزيك) ليقوم بتنفيذ عمل محدد، يمكن أن يكون مثلاً لنقل كتلة من الذاكرة، فهو يحتاج الى عنوان المقصد (على سبيل المثال (A020H)) لتمريره من برنامج (البيزيك) وتخزينه من أجل (شحنه) إلى داخل السجل DE.

سطر برنامج (البيزيك) يمكن أن يكون على الشكل التالي: 100 A = USR (&HA020)

وهذا عندئذٍ يقوم باستدعاء (روتين) لغة الآلة عند العنوان E000H وينفذ أي عمل مطلوب حتى يمر على (RETurned) والتي تعني الرجوع الى برنامج (البيزيك). العدد الصحيح أو العنوان (A020H) يكون دائماً مخزن عند العنوان F7F8 والعنوان F7F9 (الستة عشر). من أجل (شحن) هذا العنوان الى السجل DE (سجل المقصد) فإن كل ما يحتاجه برنامج لغة الآن أن يقوم بتنفيذ:

- LD DE, (0F7F8)

والسجل DE سوف يحتوي بعد ذلك على القيمة (A020H).

إذا مارغب أحد بتعديل العدد الصحيح وإرجاعه إلى برنامج لغة البيزيك عندما يعود عن طريق الأمر (RETurns) فإن عليه اضافة السطر التالي:

LD (F7F8H), DE

والقيمة الجديدة سوف توضع في المتغير A عندما يعود إلى برنامج البيزيك.

إن استخدام صف الحروف (String) كدليل يختلف اختلافاً بسيطاً في الطريقة المستخدمة فالعنوان F7F8 والعنوان F7F9 (ستة عشر) سوف لايحتويان على صف الحروف بل على عنوان الموقع الذي يصف هذه الحروف يتألف الموقع الذي يوصف هذه الحروف من ثلاثة (بايتات) حيث (البايت) الأول يشير إلى طول صف الحروف (عددها) و(البايت) الثاني والثالث يشيران إلى العنوان المخزنة به هذه الحروف،

حيث يمكن أن تستخدم على هذا الشكل:

100 B$ = "MSX"

110 A$ = USR2 (B$)

هذه المرة العنوان F663H يحتوي على القيمة 3 . العناوين F7F8 و F7F9 (ستة

عشرة) يحويان عنوان أداة الوصف على سبيل المثال 802D ، وعند هذا العنوان سوف يكون طول الحروف (عددها) والذي هو في مثالنا «3» بينما العنوان 802E والعنوان 802F سوف يحويان عنوان الموقع الحقيقي المخزن فيه صف الحروف وبترتيب معكوس بالطبع.

ـ القيم ذات الدقة الأحادية والتي تتألف من أربعة (بايتات) تخزن إبتداء من العنوان F7F6 الى العنوان F7F9 (ستة عشرة) أما القيم ذات الدقة المضاعفة، والتي تتألف من ثمانية (بايتات) وتبدأ من العنوان F7F6 الى F7F9 ستة عشر. أما القيم ذات الدقة المضاعفة التي تتألف من ثمانية (بايتات) تبدأ من العنوان F7F6 إلى F7FD (ستة عشر).

تحذير:

لايمكن معالجة عناوين المخازن (F7F6H - F7FDH) بعد عودة (الروتين) الى برنامج (البيزيك) (لايمكن إجراء عملية الـ Peek عليهم). حيث أنه لايمكن تخزينهم. فقط المتغيرات التي إستعملت في سطر الـ USR (في مثالنا السابق المتغير A$) سوف تحتوي على المعلومات. كمثال على كيفية تمرير عدد صحيح إلى (روتين) لغة الآلة (اشحن) المجمع ZEN بالطريقة العادية وأدخل هذا البرنامج القصير التالي:

برنامج

```
1 ORG 0E000H
2 LOAD 0E000H
3 LOOP:EQU 0A003H
4 LD DE,(0F7F8H)
5 JP LOOP
6 END
7 .
```

بعد تجميع البرنامج والتأكد أن كل شيء صحيح أدخل «B» للعودة إلى طور البيزيك وأدخل برنامج (البيزيك) التالي ونفذه:

برنامج

```
10 DEFUSR2=&HA000
20 A=USR2(&H1234)
30 PRINT HEX$(A)
and RUN.
```

هذا البرنامج سوف يدخل حلقة المجمع ZEN الرئيسية لذلك أدخل (GE000H) مع «RETURN» مرتين.

البرنامج القصير هذا سوف ينفذ على الفور ويعود بإشارة المجمع ZEN. الآن أدخل «X» وهذا لفحص السجلات وسوف تجد السجل DE يحوي على 1234 وهذا ما يثبت أن العدد الصحيح يمكن أن يمرر إلى (روتين) لغة الآلة.

حتى تكتشف كيف استطاع العدد الصحيح المرورو إلى (روتين) لغة الآلة.

أدخل «MF7F8H» والمحتويات سوف تظهر على شكل (34).

أدخل 35H مع «RETURN» متبوعة بنقطة «.» وأدخل «RUTERN» التي تعدل محتويات الذاكرة.

أدخل «B» للعودة إلى طور (البيزيك) والسطر 30 عندئذٍ سوف ينفذ ويطبع القيمة (الستة عشر) للمتغير A والتي سوف نراها قد تغيرت الى (1235H).

الأمر «Q» في المجمع ZEN يفيد في إظهار محتويات الذاكرة ويمكن استخدامه هنا لاكتشاف كيفية تخزين المتغيرات ذات الدقة الأحادية والمضاعفة، كما أنه إذا ما عاد التحكم الى (البيزيك) فإن منطقة التخزين سوف تخرب.

يقوم هذا الفصل بعرض لبعض (الروتينات) المفيدة الموجودة في الذاكرة ROM والتي يمكن أن تستخدم من خلال برامج لغة الآلة التي تكتبها بنفسك. بعد هذه (الروتينات) قد استخدمت في فصول سابقة من هذا الكتاب وبعضها الآخر يمكن أن يكون متقدماً حيث لا مجال لاستخدامه في الوقت الحاضر.

بداية الذاكرة ROM تحتوي على جدول القفزات (لروتينات) متعددة، بعض هذه (الروتينات) المعروفة طبعت في هذا الجدول لتساعدنا إذا رغبنا بفك رموز لغة الآلة إلى لغة التجميع لقسم محدد من الذاكرة، ولكن استدعاء الموقع المناسب في

الجدول هو الشيء الطبيعي الذي نحتاجه بالاضافة الى تزويدنا بمعرفة أي السجلات التي يجب أن (تشحن) بالمعلومات المناسبة قبل إجراء عملية الاستدعاء (CALL).

إن كل (روتين) موجود في هذا الجدول يحمل اسم مختصر (مصطلح)، أو عنوان بطول ستة أحرف كما هو مستخدم في مواصفات نظام الـ MSX.

إليك الآن قائمة هذا الجدول والتي تحتوي على أربعة أعمدة وهي:

Addr	Jump	Name	Function
العنوان	القفز	الاسم	الوظيفة
0000	02D7	CHKRAM	ـ (البايت) الأول تعطل الانقطاعات ويقفز هذا الأمر إلى 02D7 يختبر الذاكرة RAM (ذاكرة الوصول العشوائية) ومجموعة الشقوق (Slots) من أجل منطقة القيادة يتبع ذلك عنوان جدول مولد الحرف وأيضاً البوابات لمعالج الاظهار المرئي من أجل القراءة والكتابة.
0008	2683	SYNCHR	ـ يستدعى هذا (الروتين) عن طريق الأمر (RST8). يقوم بإختبار الحرف الحالي المشار إليه في السجل HL فإذا كان هو الحرف المطلوب يعود إلى (GHRGTR) مولد الحروف وإلا سوف يعطي رسالة خطأ تحوي (Syn taxe error). الحرف الذي يجب أن يختبر يجب أن يكون البايت التالية بعد الـ Rst. علم الباقي يجهز إذ كان الحرف رقماً وعلم الصفر يجهز إذا صادف نهاية الجملة. السجلات HL و AF تتغير.
0010	2686	CHRGTR	ـ ينفذ عن طريق RsT2. يستخدم مفسر لغة (البيزيك) هذا (الروتين) ليقوم باحضار الحرف التالي من نص (البيزيك)، السجل HL يشير إلى

الحرف المراد إحضاره، السجل A يحوي الحرف، علم الباقي يجهز إذا كان الحرف رقماً وعلم الصفر يجهز إذا صادف نهاية الجملة. السجلات HL و AF تتغير.

`0018 1B45 OUTDO`
ـ ينفذ هذا (السروتين) عن طريق استدعاء (RST 18H) لنقل محتويات الاخراج للسجل A إلى الوحدة المحيطية الحالية، الشاشة، الطابعة، . . . الخ.

`0020 146A DCOMPR`
ـ يقارن السجل HL مع السجل DE ويجهزعلم الصفر إذا كان هناك تطابق. السجل AF يتغير.

`0038 0C3C KEYINT`
ـ ينفذ إجراءات الإعاقة لمعدات (الكومبيوتر) خمسين مرة في الثانية. لا يتم تغيير أي شيء في السجلات.

`0041 0577 DISSCR`
ـ يعطل الشاشة وتصبح بيضاء. السجل BC و AF يتغيران.

`0044 0570 ENASCR`
ـ يعيد فعالية الشاشة بعد تعطيلها مع اظهار ما كان موجود سابقاً عليها. السجلات BC و AF تتغير.

`0047 057F WRTVDP`
ـ يقوم بكتابة المعلومات الى سجل معالج العرض المرئي (VDP) السجلات BC و AF تتغير.

`004A 07D7 RDVRM`
ـ يقرء ذاكرة العرض (VRAM) المشار إليها بالسجل HL، يعيد المعلومات إلى السجل A. السجل AF يتغير.

`004D 07CD WRTVRM`
ـ يكتب على ذاكرة العرض (VRAM) المشار إليها بالسجل HL المعلومات التي في السجل A, السجل AF يتغير.

`0050 07EC SETRD`
ـ يجهز معالج العرض المرئي (VDP) للقراءة. السجل AF يتغير.

`0053 07DF SETWRT`
ـ يجهز معالج العرض المرئي (VDP) للكتابة. السجل AF يتغير.

`0056 0815 FILVRM`
ـ يقوم بملأ ذاكرة العرض (VRAM) إبتداء من العنوان الموجود في السجل HL بالمعلومات التي في السجل A وبطول يحدده السجل BC. السجلات BC و AF يتغيران.

`0059 070F LDIRMV`
ـ يرحل كتلة من ذاكرة العرض (VRAM) إلى الذاكرة. عنوان المصدر لذاكرة العرض في السجل HL وعنوان المقصد في السجل DE وطول الكتلة في السجل BC. جميع السجلات تتغير.

`005C 0744 LDIRVM`
ـ يرحل كتلة من الذاكرة إلى ذاكرة العرض (VRAM). عنوان المصدر للذاكرة. في السجل HL وعنوان المقصد في السجل DE وطول الكتلة في السجل BC. جميع السجلات تتغير.

`005F 084F CHGMOD`
ـ يقوم بتحضير الشاشة طبقاً للقيمة الموجودة في السجل A (موديل 3-2-1-0). يخزن السجل A عند العنوان (FCAFH). جميع السجلات تتغير.

`0062 07F7 CHGCLR`
ـ يغير لون الشاشة طبقاً للون المحدد في: العنوان F3F9 للون الحرف (FORCLR) والعنوان F3E9 للون الأرضية (BAKCLR) والعنوان F3EB للون الاطار (BDRCLR). جميع السجلات تتغير.

`0066 1398 NMI`
ـ ينجز إجراء الانقطاع الذي ليس بقابل للحجب. السجلات لا تتغير.

0069 06A8 CLRSPR	ـ يجهز جميع الأشباح . النماذج للأشباح تجهز بصفر.
006C 050E INITXT	ـ يجهز الشاشة لنص الكتابة (الشاشة 0) (Screen 0) ومجموعة معالج العرض المرئي (VDP). جميع السجلات تتغير.
006F 0538 INIT32	ـ يجهز الشاشة (موديل) 1 (Screen 1) ومجموعة معالج العرض المرئي (VDP) جميع السجلات تتغير .
0072 05D2 INIGRP	ـ يجهز الشاشة موديل 2 (Screen 2) ومجموعة معالج العرض المرئي (VDP) جميع السجلات تتغير .
0075 061F INIMLT	ـ يجهز الشاشة موديل 3(Screen 3) ومجموعة معالج العرض المرئي (VDP) جميع السجلات تتغير .
0078 0594 SETTXT	ـ معالج العرض المرئي VDP للشاشة 0
007B 05B4 SETT32	ـ معالج العرض المرئي VDP للشاشة 1
007E 0602 SETGRP	ـ معالج العرض المرئي VDP للشاشة 2
0081 0659 SETMLT	ـ معالج العرض المرئي للشاشة 3
0084 06E4 CALPAT	ـ يضع عنوان جدول نموذج الشبح في السجل HL . السجل A = رقم الشبح . السجلات AF- DE- HL تتغير .
0087 06F9 CALATR	ـ يضع عنوان جدول مميز الشبح في السجل HL . رقم الشبح في السجل A . السجلات AF- DE- HL تتغير .
008A 0704 GSPSIZ	ـ يضع حجم الشبح الحالي في السجل A (عدد البايتات) يجهز علم الباقي إذا كان الشبح 16 * 16 وإلا يعيد تجهيزه . السجل AF يتغير .

008D 1510 GRPPRT	ـ يطبع الحرف الذي في السجل A على شاشة الرسومات (GRAPHIC).
0090 04BD GICINI	ـ يعطي قيمة ابتدائية للسجل PSG. جميع السجلات تتغير
0093 1102 WRTPSG	ـ يكتب المعلومات التي في السجل E الى السجل PSG .
0096 110E RDPSG	ـ يقرأ المعلومات من السجل PSG في السجل A . ويعود بالمعلومات من السجل A . جميع السجلات تتغير .
0099 11C4 STRTMS	ـ يراقب ويبدأ خلفية الموسيقا .
009C 0D6A CHSNS	ـ يراقب لوحة المفاتيح فيما إذا ضغط أحد المفاتيح . يجهز العلم «Z» إذا كان هناك مفتاح موجود في منطقة التخزين . السجل AF يتغير .
009F 10CB CHGET	ـ يقوم بالانتظار حتى يضغط مفتاح ليعود برمز الآسكي له ويضعه في السجل A والسجل AF يتغير .
00A2 08BC CHPUT	ـ يقوم بإخراج محتويات السجل A إلى الشاشة .
00A5 085D LPTOUT	ـ يقوم بإخراج محتويات السجل A إلى الطابعة . يجهزة علم الباقي بعد الانتهاء . السجل F يتغير .
00A8 0884 LPTSTT	ـ يفحص حالة الطابعة . يعود بالقيمة FF hex إلى داخل السجل A والعلم «Z» يعاد تجهيزه من جديد إذا كانت الطابعة جاهزة . يوضع 0 في السجل A والعلم «Z» يكون فيه «1» أي مجهزة إذا كانت الطابعة غير جاهزة .

00AE 23BF PINLIN	ـ يخزن سطر إدخال من لوحة المفاتيح في منطقة التخزين، يحدد نهاية السطر طبعاً عن ضغط المفتاح «RETURN» ، يضع بداية منطقة التخزين في السجل HL . علم الباقي يجهز إذا ضغط المفتاح STOP . جميع السجلات تتغير .
00B7 046F BREAKX	ـ يراقب المفتاحان CTRL/ STOP إذا ضغطا معاً . على الباقي يجهز اذا تم ذلك . السجل AF يتغير .
00C0 1113 BEEP	ـ لصوت الجرس .
00C3 0848 CLS	ـ ينظف الشاشة إذا كان علم الصفر «Z» مجهزة (SET).
00C6 088E POSIT	ـ لتحديد موقع النقطة المضيئة (CURSOR). السجل H = العمود السجل L = السطر. السجل AF يتغير .
00C9 0B26 FNKSB	ـ يراقب مفاتيح الوظائف فإذا كانت بحالة «ON» أي جاهزة يقوم بإظهارها على الشاشة وإذا لم تكن كذلك لا يقوم بأي شيء . جميع السجلات تتغير .
00CC 0B15 ERAFNK	ـ يقوم بإخفاء مفاتيح الوظائف على الشاشة «OFF» . جميع السجلات تتغير .
00CF 0B2B DSPFNK	ـ يعيد اظهار مفاتيح الوظائف على الشاشة جميع السجلات تتغير .
00D2 083B TOTEXT	ـ يعيد الشاشة الى نموذج نص الكتابة . جميع السجلات تتغير .
0156 0468 KILBUF	ـ ينظف منطقة تخزين لوحة المفاتيح السجل HL يتغير .

العناوين F380H فما فوق مخصصة كمناطق تخزين تعالج من قبل الذاكرة (ROM) أو من البرامج التي تكتبها بنفسك في الذاكرة (RAM).

القائمة التالية تحوي على أسماء المناطق الهامة وشائعة الاستعمال في نظام الـ MSX مع عدد البايتات والغرض منها .

على سبيل المثال : طول السطر الحالي في الشاشة (موديل) 0 (Screen 0) يكون في العنوان F3AEH ، وعادة يكون مساوي لي (25H) (37 عشري) .

لمراقبة محتويات هذاالموقع في الذاكرة نستطيع إدخال سطر (البيزيك) التالي : (PEEK (F&HF3AE ? أو من خلال المجمع QF3AEH : ZEN .

F39A USRTAB 20	ـ هذه العناوين مخصصة لعشروظائف للمستخدمين (0 (USR) إلى 9) . جميع هذه العناوين تحتوي على القيمة (475A) حتى تنفذ التعليمة (DEF USR) وهذه القيمة تشحن الخطأ رقم (5) في علم الخطأ .
F3AE LINL40 1	ـ عرض السطر في الشاشة (موديل) 0 (Screen 0)
F3AF LINL32 1	ـ عرض السطر في الشاشة (موديل) 1 (Screen 1)
F3B0 LINLEN 1	ـ طول السطر .
F3B2 CLMLST 1	ـ عدد الأسطر على الشاشة (شاشة «0» (Screen 0)
F3B3 TXTNAM 2	ـ بداية جدول عنوان الاسم (0000H)
F3B5 TXTCOL 2	ـ بداية جدول عنوان اللون (غير مستخدم)
F3B7 TXTCGP 2	ـ بداية جدول مولد الأحرف (0800H)
F3BB TXTPAT 2	ـ بداية جدول مولد نموذج الشبح (غير مستخدم)
	ـ الشاشة (موديل) «1» (Screen 1)
F3BD T32NAM 2	ـ بداية جدول عنوان الاسم . (1800H)
F3BF T32COL 2	ـ بداية جدول عنوان اللون . (2000H)
F3C1 T32CGP 2	ـ بداية جدول مولد الحرف . (0000H)
F3C3 T32ATR 2	ـ بداية جدول المميز . (1B00H)
F3C5 T32PAT 2	ـ بداية جدول مولد نموذج الشبح . (3800H)

الوصف	العنوان	الاسم	الطول
ـ الشاشة (موديل) 2 (Screen 2)			
ـ بداية جدول عنوان الاسم . (1800H)	F3C7	GRPNAM	2
ـ بداية جدول عنوان اللون . (2000H)	F3C9	GRPCOL	2
ـ بداية جدول مولد الحرف . (0000H)	F3CB	GRPCGP	2
ـ بداية جدول مولد المميز . (1B00 H)	F3CD	GRPATR	2
ـ بداية جدول مولد نموذج الشبح (3800H)	F3CF	GRPPAT	2
ـ الشاشة موديل 3 (Screen 3)			
ـ بداية جدول عنوان الأسم (0800H)	F3D1	MLTNAM	2
ـ بداية جدول عنوان اللون (غير مستخدم)	F3D3	MLTCOL	2
ـ بداية جدول مولد الحرف (0000H)	F3D5	MLTCGP	2
ـ بداية جدول المميز . (1B00H)	F3D7	MLTATR	2
ـ بداية جدول مولد نموذج الشبح (3800H)	F3D9	MLTPAT	2
ـ مفتاح تحويل (التكتة) 0 = ايقاف (تكتكة) المفاتيح وأي قيمة أخرى تعيد (التكتكة) .	F3DB	CLIKSW	1
ـ موقع سطر النقطة المضيئة y (Cursor)	F3DC	CSRY	1
ـ موقع عمود النقطة المضيئة X (Cursor)	F3DD	CSRX	1
ـ تحويل مفتاح إظهار الوظائف على الشاشة 0 = إيقاف	F3DE	CNSDFG	1
قيم سجل معالج العرض المرئي VDP			
ـ مخازن معالج العرض المرئي VDP من 0 إلى 7 .	F3DF to F3E6		8
ـ حالة سجل مخازن معالج العرض المرئي VDP .	F3E7	STATFL	1
ـ لون الحرف .	F3E9	FORCLR	1
ـ لون الأرضية .	F3EA	BAKCLR	1
ـ لون الأطار .	F3EB	BDRCLR	1
ـ مخزن منطقة الادخال .	F55E	BUF	256
ـ أعلى موقع في الذاكرة .	F672	MEMSIZ	2

الفصل الخامس
باحث البايت
Byte searcher

(يشحن) هذا البرنامج المساعد من المجمع Zen ويضيف بشكل بسيط (روتين) البحث عن البايت والذي يفيد عند فك لغة الآلة إلى لغة التجميع لقسم من الذاكرة، يمكن البحث أيضاً عن عنوان (بايتين) أو صف من الحروف .

البحث عن (بايتين) .
Two Byte Search

إن (روتين) لوحة المفاتيح في المجمع ZEN يبدأ من العنوان A742H ، فرض أننا نريد أن نكتشف أين وكيف كان يشار إلى (روتين) لوحة المفاتيح في داخل منطقة الذاكرة التي يشغلها المجمع ZEN .

لندخل : YA742H .

تأكد أن العنوان قد أدخل بشكل صحيح . ليس كما هو موجود في الذاكرة (البايت) الدنيا أولاً ، حيث أن (روتين) البحث قد عدل من أجل هذا .

الأمر «Y» قد استخدم كمعظم الاحرف الاخرى قبل الآن كمساعد ، يمكن أيضاً أن يعدل هذا الحرف في السطر 11 إلى أحرف صغيرة مثل «...» تستعمل خلافاً لذلك .

سوف تظهر رسالة لادخال عنوان البداية «START» متبوعاً بـ «H» فإذا كان المجمع ZEN هو المراد البحث فيه فندخل عنوان أول موقع في الذاكرة للمجمع ZEN : A000H . ومن المنطقي أن تكون الرسالة الثانية التي تظهر على الشاشة ...

النهاية «END» ففي مثالنا هذا هو آخر عنوان للمجمع BB 5 CH : ZEN.

الرسالة الأخيرة التي تظهر على الشاشة هي اختيارية «OPTION» من أجل إظهار المواقع على الشاشة، فيمكن أن ندخل مثلاً «V» .

سوف تظهر لنا الشاشة بعد ذلك ما يلي :

أي : مصادفة A742H

بين العنوان A000 إلى BB 5C

في الموقع : A7F4 A92A

```
Occurences of A742H
between:-A000 and BB5C

A7F4 A92A
ZEN
```

البحث عن صف حروف.

String Search

يمكن البحث عن صف حروف (String) ما وذلك بعد وضعه داخل قوسين علويين كما يلي : «OK» Y

لكي يتم البحث عن رسالة «OK» .

لايجاد موقع هذه الرسالة في ذاكرة (الروم) «ROM» يجب ادخال عنوان البداية والذي هو «0000» أو «0» وعنوان النهاية هو أعلى عنوان في الذاكرة ROM والذي هو «8000H» .

الشاشة سوف تظهر لنا بعد ذلك :

مصادفة الرسالة «OK»

بين العنوان 0000 و 80000

```
Occurences of "Ok"
between:-0000 and 8000

3FD7
ZEN
```

الموقع 3FD7 . والذي هو موقع وجود الرسالة «OK» في الذاكرة ROM. البحث عن (بايتين) يمكن عندئذٍ أن يستخدم لاكتشاف أي المناطق للذاكرة ROM تعالج الرسالة «OK» وذلك بالبحث عن العنوان 3FD7 بين العنوان «0» والعنوان 8000 H .

حيث يمكن أن تظهر لنا نتيجة البحث المناطق التي تشير إلى هذه الرسالة والتي هي :

412F- 53FB - 7012

إن باحث (البايت) يمكن أن يعالج عدة (روتينات) داخل المجمع ZEN وذلك فقط باستدعاء (الروتين) إلى الخارج عند العنوان 0020H حيث كان (روتين) الذاكرة ROM لمقارنة السجل HL مع السجل DE ، وبعد ذلك تطبع جميع الروتينات في حقل التعليق لكي يمكن تدقيقها مع كتيب التشغيل للمجمع ZEN الذي لديك .

يمكن حفظ هذا البرنامج على شكل ملف (آسكي) ، حيث بشكل بسيط ندخل «W» وبعد ذلك ندخل اسم الملف، ويمكن بعد ذلك أن يشحن هذا البرنامج من جديد ويجمع عندما نحتاج لتسهيلات إضافية للبحث عن البايت من أجل فك لغة الآلة إلى لغة التجميع .

ملاحظة :

بعد إدخال الرمز، (والشحن) من الشريط من الضروري تجميع برنامج باحث البايت قبل إجراء التعديل (للبايتات) الثلاثة عند العنوان A251H حيث أن هذه المنطقة تكون في داخل الحلقة الرئيسية للمجمع ZEN والقفز سيتم إلى العنوان E000H للكشف فيما اذا تم ضغط المفتاح «Y» .

فإذا كان البرنامج لم يجمع فإنه سوف لن يكون في العنوان E000H حيث يمكن للمجمع ZEN في هذه الحالة أن يَخرب ويضيع البرنامج .

```
1                     ;BYTESEARCHER
2                     ;AFTER ASSEMBLY ALTER
3                     ;ZEN BY:-  M0A251H
4                     ;and enter these 3 bytes
5                     ;0C3H  00  0E0H
6                     ;
7                               ORG  0E000H
8                               LOAD 0E000H
9                     ;
10 E000 CAA5A3        EXTRA:    JP   Z,0A3A5H       ;Orig routine
11 E003 FE59                    CP   "Y"            ;For Bytesearcher
12 E005 2803                    JR   Z,BYTSCH       ;It's what we want
13                    ;
14                    ;New commands go here
15                    ;
16 E007 C354A2                  JP   0A254H         ;Back to Zen
17                    ;
18 E00A 118AA1        BYTSCH:   LD   DE,0A18AH      ;(TBUFF+1)
19 E00D 21D6E0                  LD   HL,SCHSTR      ;Store Input
20 E010 010000                  LD   BC,0           ;String counter
21                    ;
22                    ;Transfer the string
23                    ;
24 E013 1A            TRSTR:    LD   A,(DE)
25 E014 77                      LD   (HL),A
26 E015 23                      INC  HL
27 E016 13                      INC  DE
28 E017 03                      INC  BC
29 E018 FE0D                    CP   0DH            ;Return?
30 E01A 20F7                    JR   NZ,TRSTR       ;No-keep transfering
31                    ;
32                    ;Transfer complete-Check that
33                    ;something is there
34                    ;
35 E01C 78                      LD   A,B
36 E01D 0D                      DEC  C              ;Don't count 'CR'
37 E01E B1                      OR   C
```



```
38 E01F CAD5A8                  JP   Z,0A8D5H       ;Error, so 'HUH?'
39                    ;
40                    ;Now get START/STOP parameters
41                    ;
42 E022 CDC5A8                  CALL 0A8C5H         ;'STARTSTOP'
43 E025 2B                      DEC  HL
44 E026 19                      ADD  HL,DE
45 E027 ED532CA1                LD   (0A12CH),DE    ;=START
46 E02B 222EA1                  LD   (0A12EH),HL    ;=STOP
47 E02E CD39AB                  CALL 0AB39H
48                    ;
49                    ;Print title
50                    ;
51 E031 21E0E0                  LD   HL,MSG1
52 E034 CDDCA7                  CALL 0A7DCH         ;ZEN "STR1"
53 E037 21D6E0                  LD   HL,SCHSTR
54 E03A CDDCA7                  CALL 0A7DCH
55 E03D 21EFE0                  LD   HL,MSG2
56 E040 CDC4A6                  CALL 0A6C4H
57 E043 CDDCA7                  CALL 0A7DCH
58 E046 2A2CA1                  LD   HL,(0A12CH)    ;Start of Data
59 E049 CD95A9                  CALL 0A995H         ;ZEN "WORDSP"
60 E04C 21F9E0                  LD   HL,MSG3
61 E04F CDDCA7                  CALL 0A7DCH
62 E052 2A2EA1                  LD   HL,(0A12EH)    ;End of Data
63 E055 CD95A9                  CALL 0A995H
64 E058 CDC4A6                  CALL 0A6C4H         ;ZEN "CRLF"
65 E05B CDC4A6                  CALL 0A6C4H         ;another CRLF
66                    ;
67                    ;Check for H or Quote at end
68                    ;
69 E05E 21D6E0                  LD   HL,SCHSTR
70 E061 0600                    LD   B,0            ;Counter for convert
71 E063 7E            FENDS:    LD   A,(HL)         ;Find string end
72 E064 FE0D                    CP   0DH
73 E066 2804                    JR   Z,COMP
74 E068 23                      INC  HL
75 E069 04                      INC  B
76 E06A 18F7                    JR   FENDS
```

```
116 E0AE FE0D                   CP    0DH
117 E0B0 2807                   JR    Z,FOUND
118 E0B2 BE                     CP    (HL)
119 E0B3 20E3                   JR    NZ,FINDIT
120 E0B5 13                     INC   DE
121 E0B6 23                     INC   HL
122 E0B7 18F4                   JR    LOOKIT
123                 ;
124 E0B9 E1         FOUND:      POP   HL           ;Address this srch
125 E0BA E5                     PUSH  HL           ;Restack it
126 E0BB CD95A9                 CALL  0A995H
127 E0BE 3AD5E0                 LD    A,(COUNT)
128 E0C1 3C                     INC   A
129 E0C2 FE05                   CP    5
130 E0C4 32D5E0                 LD    (COUNT),A
131 E0C7 20CF                   JR    NZ,FINDIT
132 E0C9 AF                     XOR   A
133 E0CA 32D5E0                 LD    (COUNT),A
134 E0CD CDC4A6                 CALL  0A6C4H
135 E0D0 18C6                   JR    FINDIT
136                 ;
137 E0D2 C300A0                 JP    0A000H
138 E0D5 00         COUNT:      DB    0
139                 SCHSTR:     DS    10
140 E0E0 4F636375 MSG1:         DB    "Occurences of ",0DH
140 E0E4 72656E63
140 E0E8 6573206F
140 E0EC 66200D
141 E0EF 62657477 MSG2:         DB    "between:-",0DH
141 E0F3 65656E3A
141 E0F7 2D0D
142 E0F9 616E6420 MSG3:         DB    "and ",0DH
142 E0FD 0D
143                             END
```



```
77                  ;
78 E06C 2B          COMP:       DEC   HL           ;Back-up to 'H' or "
79 E06D 7E                      LD    A,(HL)
80 E06E FE22                    CP    22H          ;It's a quote string
81 E070 2816                    JR    Z,SEEK
82 E072 FE48                    CP    "H"
83 E074 C2D5A8                  JP    NZ,0A8D5H    ;Not hex
84 E077 23                      INC   HL           ;Back to end
85 E078 11D6E0                  LD    DE,SCHSTR
86 E07B CDDAA8                  CALL  0A8DAH       ;ZEN convert routine
87 E07E 22D7E0                  LD    (SCHSTR+1),HL
88 E081 3E0D                    LD    A,0DH
89 E083 32D9E0                  LD    (SCHSTR+3),A
90 E086 1802                    JR    FIND
91                  ;
92 E088 360D        SEEK:       LD    (HL),0DH
93 E08A 2A2CA1      FIND:       LD    HL,(0A12CH)
94 E08D ED5B2EA1                LD    DE,(0A12EH)
95 E091 2B                      DEC   HL
96 E092 D5                      PUSH  DE
97 E093 E5                      PUSH  HL
98                  ;
99 E094 AF                      XOR   A
100                 ;
101 E095 32D5E0                 LD    (COUNT),A
102 E098 E1         FINDIT:     POP   HL
103 E099 D1                     POP   DE
104 E09A 23                     INC   HL
105 E09B D5                     PUSH  DE
106 E09C E5                     PUSH  HL
107 E09D CD2000                 CALL  0020H        ;ROM Compare HL'DE
108 E0A0 2008                   JR    NZ,LOOK
109 E0A2 E1                     POP   HL
110 E0A3 D1                     POP   DE
111 E0A4 CDC4A6                 CALL  0A6C4H       ;Finished so CRLF
112 E0A7 C300A0                 JP    0A000H       ;Back to ZEN
113                 ;
114 E0AA 11D7E0     LOOK:       LD    DE,SCHSTR+1
115 E0AD 1A         LOOKIT:     LD    A,(DE)
```

الملحق

Introduction

جدول تحويل القيمة ستة عشر إلى رمز العملية .

HEX To opcode Conversion table

إن هذا الجدول يساعدنا عندما نكون نعرف القيمة (الستة عشر) ونرغب في معرفة رمز العملية لهذه القيمة مع عدد البايتات التي تتبعها .

عندما يحاول أحد تحويل القيمة العشرية التي في تعليمة الـ DATA في لغة (البيزيك) إلى رمز عملية والمعاملات (الأرقام) تبدأ بالتأكيد من البايت الأولى في الروتين وإلا فسوف نحصل على معلومات مغلوطة وخاطئة .

على سبيل المثال لنأخذ البرنامج رقم «1» في الفصل الأول . البايت الأولى في تعليمة الـ DATA تحوي القيمة العشرية (62) ، لنحول هذه القيمة العشرية إلى قيمة (ستة عشر) فسوف نجدها (3E hex). لننظر الآن إلى الجدو التالي لنجد ماذا يقابل القيمة (3E) hex من رمز عملية . نجدها تقابل الأمر: (LD A, nn) والذي يعني (اشحن) السجل A بالقيمة التي في (البايت) التالي والتي تكون هي (66 عشري) أو (42 hex) (ستة عشر) . لنتابع الآن القيمة الثالثة في سطر الـ DATA والتي تكون 33 (عشري) أو (21 hex) (ستة عشر). عند التدقيق بالجدول نجد أن هذه القيمة تقابل الأمر: (LD HL, aa bb) والذي يعني (شحن) (البايتين) التاليتين إلى السجل HL وهكذا . . . .

إذا أراد أحد أن يبدأ بالتحويل من مكان خاطىء مثلاً البايت الثاني أو البايت الثالث فوجد القيمة العشرية 66 والتي تساوي 42 (ستة عشر) فنظر إلى الجدول ليرى ما يقابل هذه القيمة فوجد الأمر التالي: (LD B,D) . إن رمز العملية صحيح ولكن بالنتيجة

سوف يعطيها معلومات خاطئة لأننا لم نبدأ بالتحويل من البايت الأولى لذلك لم ندرك ماذا تعنيه بالضبط البايت الثالثة.

لذلك من الجوهري أن نبدأ بالتحويل من البايت الأولى من سطر الـ DATA .

ـ في هذا الجدول nn تساوي قيمة بايت واحد في المجال:

(00h إلى FF h) (ستة عشر).

(0 الى 255 dec) عشري

ـ bbaa تساوي قيمة (بايتين) في المجال السابق نفسه.

الجدول

00	NOP	0C	INC C
01 bb aa	LD BC,aabb	0D	DEC C
02	LD (BC),A	0E nn	LD C,nn
03	INC BC	0F	RRCA
04	INC B	10 nn	DJNZ nn
05	DEC B	11 bb aa	LD DE,aabb
06 nn	LD B,nn	12	LD (DE),A
07	RLCA	13	INC DE
08	EX AF,AF'	14	INC D
09	ADD HL,BC	15	DEC D
0A	LD A,(BC)	16 nn	LD D,nn
0B	DEC BC	17	RLA
18 nn	JR nn	3D	DEC A
19	ADD HL,DE	3E nn	LD A,nn
1A	LD A,(DE)	3F	CCF
1B	DEC DE	40	LD B,B
1C	INC E	41	LD B,C
1D	DEC E	42	LD B,D
1E nn	LD E,nn	43	LD B,E
1F	RRA	44	LD B,H
20 nn	JR NZ,nn	45	LD B,Ln
21 bb aa	LD HL,aabb	46	LD B,(HL)
22 bb aa	LD (aabb),HL	47	LD B,A
23	INC HL	48	LD C,B

24	INC H	49	LD C,C
25	DEC H	4A	LD C,D
26 nn	LD H,nn	4B	LD C,E
27	DAA	4C	LD C,H
28 nn	JR Z,nn	4D	LD C,L
29	ADD HL,HL	4E	LD C,(HL)
2A bb aa	LD HL,(nn)	4F	LD C,A
2B	DEC HL	50	LD D,B
2C	INC L	51	LD D,C
2D	DEC L	52	LD D,D
2E nn	LD L,nn	53	LD D,E
2F	CPL	54	LD D,H
30 nn	JR NC,nn	55	LD D,L
31 bb aa	LD SP,aabb	56	LD D,(HL)
32 bb aa	LD (aabb),A	57	LD D,A
33	INC SP	58	LD E,B
34	INC (HL)	59	LD E,C
35	DEC (HL)	5A	LD E,D
36 nn	LD (HL),nn	5B	LD E,E
37	SCF	5C	LD E,H
38 nn	JR C,nn	5D	LD E,L
39	ADD HL,SP	5E	LD E,(HL)
3A bb aa	LD A,(aabb)	5F	LD E,A
3B	DEC SP	60	LD H,B
3C	INC A	61	LD H,C
62	LD H,D	85	ADD A,L
63	LD H,E	86	ADD A,(HL)
64	LD H,H	87	ADD A,A
65	LD H,L	88	ADC A,B
66	LD H,(HL)	89	ADC A,C
67	LD H,A	8A	ADC A,D
68	LD L,B	8B	ADC A,E
69	LD L,C	8C	ADC A,H
6A	LD L,D	8D	ADC A,L
6B	LD L,E	8E	ADC A,(HL)
6C	LD L,H	8F	ADC A,A
6D	LD L,L	90	SUB B
6E	LD L,(HL)	91	SUB C
6F	LD L,A	92	SUB D

70	LD (HL),B	93	SUB E
71	LD (HL),C	94	SUB H
72	LD (HL),D	95	SUB L
73	LD (HL),E	96	SUB (HL)
74	LD (HL),H	97	SUB A
75	LD (HL),L	98	SBC A,B
76	HALT	99	SBC A,C
77	LD (HL),A	9A	SBC A,D
78	LD A,B	9B	SBC A,E
79	LD A,C	9C	SBC A,H
7A	LD A,D	9D	SBC A,L
7B	LD A,E	9E	SBC A,(HL)
7C	LD A,H	9F	SBC A,A
7D	LD A,L	A0	AND B
7E	LD A,(HL)	A1	AND C
7F	LD A,A	A2	AND D
80	ADD A,B	A3	AND E
81	ADD A,C	A4	AND H
82	ADD A,D	A5	AND L
83	ADD A,E	A6	AND (HL)
84	ADD A,H	A7	AND A
A8	XOR B	CB 00	RLC B
A9	XOR C	CB 01	RLC C
AA	XOR D	CB 02	RLC D
AB	XOR E	CB 03	RLC E
AC	XOR H	CB 04	RLC H
AD	XOR L	CB 05	RLC L
AE	XOR (HL)	CB 06	RLC (HL)
AF	XOR A	CB 07	RLC A
B0	OR B	CB 08	RRC B
B1	OR C	CB 09	RRC C
B2	OR D	CB 0A	RRC D
B3	OR E	CB 0B	RRC E
B4	OR H	CB 0C	RRC H
B5	OR L	CB 0D	RRC L
B6	OR (HL)	CB 0E	RRC (HL)
B7	OR A	CB 0F	RRC A
B8	CP B	CB 10	RL B
B9	CP C	CB 11	RL C

BA	CP D	CB 12	RL D
BB	CP E	CB 13	RL E
BC	CP H	CB 14	RL H
BD	CP L	CB 15	RL L
BE	CP (HL)	CB 16	RL (HL)
BF	CP A	CB 17	RL A
C0	RET NZ	CB 18	RR B
C1	POP BC	CB 19	RR C
C2 bb aa	JP NZ,aabb	CB 1A	RR D
C3 bb aa	JP aabb	CB 1B	RR E
C4 bb aa	CALL NZ,aabb	CB 1C	RR H
C5	PUSH BC	CB 1D	RR L
C6 nn	ADD A,nn	CB 1E	RR (HL)
C7	RST 00	CB 1F	RR A
C8	RET Z	CB 20	SLA B
C9	RET	CB 21	SLA C
CA bb aa	JP Z,aabb	CB 22	SLA D
CB 23	SLA E	CB 46	BIT 0,(HL)
CB 24	SLA H	CB 47	BIT 0,A
CB 25	SLA L	CB 48	BIT 1,B
CB 26	SLA (HL)	CB 49	BIT 1,C
CB 27	SLA A	CB 4A	BIT 1,D
CB 28	SRA B	CB 4B	BIT 1,E
CB 29	SRA C	CB 4C	BIT 1,H
CB 2A	SRA D	CB 4D	BIT 1,L
CB 2B	SRA E	CB 4E	BIT 1,(HL)
CB 2C	SRA H	CB 4F	BIT 1,A
CB 2D	SRA L	CB 50	BIT 2,B
CB 2E	SRA (HL)	CB 51	BIT 2,C
CB 2F	SRA A	CB 52	BIT 2,D
CB 30	SLI B	CB 53	BIT 2,E
CB 31	SLI C	CB 54	BIT 2,H
CB 32	SLI D	CB 55	BIT 2,L
CB 33	SLI E	CB 56	BIT 2,(HL)
CB 34	SLI H	CB 57	BIT 2,A
CB 35	SLI L	CB 58	BIT 3,B
CB 36	SLI (HL)	CB 59	BIT 3,C
CB 37	SLI A	CB 5A	BIT 3,D
CB 38	SRL B	CB 5B	BIT 3,E

CB 39	SRL C
CB 3A	SRL D
CB 3B	SRL E
CB 3C	SRL H
CB 3D	SRL L
CB 3E	SRL (HL)
CB 3F	SRL A
CB 40	BIT 0,B
CB 41	BIT 0,C
CB 42	BIT 0,D
CB 43	BIT 0,E
CB 44	BIT 0,H
CB 45	BIT 0,L
CB 69	BIT 5,C
CB 6A	BIT 5,D
CB 6B	BIT 5,E
CB 6C	BIT 5,H
CB 6D	BIT 5,L
CB 6E	BIT 5,(HL)
CB 6F	BIT 5,A
CB 70	BIT 6,B
CB 71	BIT 6,C
CB 72	BIT 6,D
CB 73	BIT 6,E
CB 74	BIT 6,H
CB 75	BIT 6,L
CB 76	BIT 6,(HL)
CB 77	BIT 6,A
CB 78	BIT 7,B
CB 79	BIT 7,C
CB 7A	BIT 7,D
CB 7B	BIT 7,E
CB 7C	BIT 7,H
CB 7D	BIT 7,L
CB 7E	BIT 7,(HL)
CB 7F	BIT 7,A
CB 80	RES 0,B
CB 81	RES 0,C
CB 82	RES 0,D

CB 5C	BIT 3,H
CB 5D	BIT 3,L
CB 5E	BIT 3,(HL)
CB 5F	BIT 3,A
CB 60	BIT 4,B
CB 61	BIT 4,C
CB 62	BIT 4,D
CB 63	BIT 4,E
CB 64	BIT 4,H
CB 65	BIT 4,L
CB 66	BIT 4,(HL)
CB 67	BIT 4,A
CB 68	BIT 5,B
CB 8C	RES 1,H
CB 8D	RES 1,L
CB 8E	RES 1,(HL)
CB 8F	RES 1,A
CB 90	RES 2,B
CB 91	RES 2,C
CB 92	RES 2,D
CB 93	RES 2,E
CB 94	RES 2,H
CB 95	RES 2,L
CB 96	RES 2,(HL)
CB 97	RES 2,A
CB 98	RES 3,B
CB 99	RES 3,C
CB 9A	RES 3,D
CB 9B	RES 3,E
CB 9C	RES 3,H
CB 9D	RES 3,L
CB 9E	RES 3,(HL)
CB 9F	RES 3,A
CB A0	RES 4,B
CB A1	RES 4,C
CB A2	RES 4,D
CB A3	RES 4,E
CB A4	RES 4,H
CB A5	RES 4,L

CB 83	RES 0,E
CB 84	RES 0,H
CB 85	RES 0,L
CB 86	RES 0,(HL)
CB 87	RES 0,A
CB 88	RES 1,B
CB 89	RES 1,C
CB 8A	RES 1,D
CB 8B	RES 1,E
CB AF	RES 5,A
CB B0	RES 6,B
CB B1	RES 6,C
CB B2	RES 6,D
CB B3	RES 6,E
CB B4	RES 6,H
CB B5	RES 6,L
CB B6	RES 6,(HL)
CB B7	RES 6,A
CB B8	RES 7,B
CB B9	RES 7,C
CB BA	RES 7,D
CB BB	RES 7,E
CB BC	RES 7,H
CB BD	RES 7,L
CB BE	RES 7,(HL)
CB BF	RES 7,A
CB C0	SET 0,B
CB C1	SET 0,C
CB C2	SET 0,D
CB C3	SET 0,E
CB C4	SET 0,H
CB C5	SET 0,L
CB C6	SET 0,(HL)
CB C7	SET 0,A
CB C8	SET 1,B
CB C9	SET 1,C
CB CA	SET 1,D
CB CB	SET 1,E
CB CC	SET 1,L

CB A6	RES 4,(HL)
CB A7	RES 4,A
CB A8	RES 5,B
CB A9	RES 5,C
CB AA	RES 5,D
CB AB	RES 5,E
CB AC	RES 5,H
CB AD	RES 5,L
CB AE	RES 5,(HL)
CB D2	SET 2,D
CB D3	SET 2,E
CB D4	SET 2,H
CB D5	SET 2,L
CB D6	SET 2,(HL)
CB D7	SET 2,A
CB D8	SET 3,B
CB D9	SET 3,C
CB DA	SET 3,D
CB DB	SET 3,E
CB DC	SET 3,H
CB DD	SET 3,L
CB DE	SET 3,(HL)
CB DF	SET 3,A
CB E0	SET 4,B
CB E1	SET 4,C
CB E2	SET 4,D
CB E3	SET 4,E
CB E4	SET 4,H
CB E5	SET 4,L
CB E6	SET 4,(HL)
CB E7	SET 4,A
CB E8	SET 5,B
CB E9	SET 5,C
CB EA	SET 5,D
CB EB	SET 5,E
CB EC	SET 5,H
CB ED	SET 5,L
CB EE	SET 5,(HL)
CB EF	SET 5,A

DD CB nn 46	BIT 0,(IX+nn)
DD CB nn 4E	BIT 1,(IX+nn)
DD CB nn 56	BIT 2,(IX+nn)
DD CB nn 5E	BIT 3,(IX+nn)
DD CB nn 66	BIT 4,(IX+nn)
DD CB nn 6E	BIT 5,(IX+nn)
DD CB nn 76	BIT 6,(IX+nn)
DD CB nn 7E	BIT 7,(IX+nn)
DD CB nn 86	RES 0,(IX+nn)
DD CB nn 8E	RES 1,(IX+nn)
DD CB nn 96	RES 2,(IX+nn)
DD CB nn 9E	RES 3,(IX+nn)
DD CB nn A6	RES 4,(IX+nn)
DD CB nn AE	RES 5,(IX+nn)
DD CB nn B6	RES 6,(IX+nn)
DD CB nn BE	RES 7,(IX+nn)
DD CB nn C6	SET 0,(IX+nn)
DD CB nn CE	SET 1,(IX+nn)
DD CB nn D6	SET 2,(IX+nn)
DD CB nn DE	SET 3,(IX+nn)
DD CB nn E6	SET 4,(IX+nn)
DD CB nn EE	SET 5,(IX+nn)
DD CB nn F6	SET 6,(IX+nn)
DD CB nn FE	SET 7,(IX+nn)
DD E1	POP IX
DD E3	EX (SP),IX
DD E5	PUSH IX
DD E9	JP (IX)
DD F9	LD SP,IX
DE nn	SBC A,nn
DF	RST 18
E0	RET PO
E1	POP HL
E2 bb aa	JP PO,aabb
E3	EX (SP),HL
ED 61	OUT (C),H
ED 62	SBC HL,HL
ED 67	RRD
ED 68	IN L,(C)

E4 bb aa	CALL PO,aabb
E5	PUSH HL
E6 nn	AND nn
E7	RST 20
E8	RET PE
E9	JP (HL)
EA bb aa	JP PE,aabb
EB	EX DE,HL
EC bb aa	CALL PE,aabb
ED 40	IN B,(C)
ED 41	OUT (C),B
ED 42	SBC HL,BC
ED 43 bb aa	LD (aabb
ED 44	NEG
ED 45	RETN
ED 46	IM 0
ED 47	LD I,A
ED 48	IN C,(C)
ED 49	OUT (C),C
ED 4A	ADC HL,BC
ED 4B bb aa	LD BC,(aabb)
ED 4D	RETI
ED 4F	LD R,A
ED 50	IN D,(C)
ED 51	OUT (C),D
ED 53 bb aa	LD (aabb),DE
ED 56	IM 1
ED 57	LD A,I
ED 58	IN E,(C)
ED 59	OUT (C),E
ED 5A	ADC HL,DE
ED 5B bb aa	LD DE,(aabb)
ED 5E	IM 2
ED 5F	LD A,R
ED 60	IN H,(C)
F3	DI
F4 bb aa	CALL P,aabb
F5	PUSH AF
F6 nn	OR nn

CB CD	SET 1,L
CB CE	SET 1,(HL)
CB CF	SET 1,A
CB D0	SET 2,B
CB D1	SET 2,C
CB F5	SET 6,L
CB F6	SET 6,(HL)
CB F7	SET 6,A
CB F8	SET 7,B
CB F9	SET 7,C
CB FA	SET 7,D
CB FB	SET 7,E
CB FC	SET 7,H
CB FD	SET 7,L
CB FE	SET 7,(HL)
CB FF	SET 7,A
CC bb aa	CALL Z,aabb
CD bb aa	CALL aabb
CE nn	ADC A,nn
CF	RST 08
D0	RET NC
D1	POP DE
D2 bb aa	JP NC,aabb
D3 nn	OUT (nn),A
D4 bb aa	CALL NC,aabb
D5	PUSH DE
D6 nn	SUB nn
D7	RST 10
D8	RET C
D9	EXX
DA bb aa	JP C,aabb
DB nn	IN A,(nn)
DC bb aa	CALL C,nn
DD 09	ADD IX,BC
DD 19	ADD IX,DE
DD 21 bb aa	LD IX,aabb
DD 22 bb aa	LD (aabb),IX
DD 23	INC IX
DD 29	ADD IX,IX
DD 2A bb aa	LD IX,(aabb)

CB F0	SET 6,B
CB F1	SET 6,C
CB F2	SET 6,D
CB F3	SET 6,E
CB F4	SET 6,H
DD 2B	DEC IX
DD 34 nn	INC (IX+nn)
DD 35 nn	DEC (IX+nn)
DD 36 nn n1	LD (IX+nn),n1
DD 39	ADD IX,SP
DD 46 nn	LD B,(IX+nn)
DD 4E nn	LD C,(IX+nn)
DD 56 nn	LD D,(IX+nn)
DD 5E nn	LD E,(IX+nn)
DD 66 nn	LD H,(IX+nn)
DD 6E nn	LD L,(IX+nn)
DD 70 nn	LD (IX+nn),B
DD 71 nn	LD (IX+nn),C
DD 72 nn	LD (IX+nn),D
DD 73 nn	LD (IX+nn),E
DD 74 nn	LD (IX+nn),H
DD 75 nn	LD (IX+nn),L
DD 77 nn	LD (IX+nn),A
DD 7E nn	LD A,(IX+nn)
DD 86 nn	ADD A,(IX+nn)
DD 8E nn	ADC A,(IX+nn)
DD 96 nn	SUB (IX+nn)
DD 9E nn	SBC A,(IX+nn)
DD A6 nn	AND (IX+nn)
DD AE nn	XOR (IX+nn)
DD B6 nn	OR (IX+nn)
DD BE nn	CP (IX+nn)
DD CB nn 06	RLC (IX+nn)
DD CB nn 0E	RRC (IX+nn)
DD CB nn 16	RL (IX+nn)
DD CB nn 1E	RR (IX+nn)
DD CB nn 26	SLA (IX+nn)
DD CB nn 2E	SRA (IX+nn)
DD CB nn 36	SLI (IX+nn)
DD CB nn 3E	SRL (IX+nn)

ED 69	OUT (C),L
ED 6A	ADC HL,HL
ED 6F	RLD
ED 70	IN F,(C)
ED 72	SBC HL,SP
ED 73 bb aa	LD (aabb),SP
ED 78	IN A,(C)
ED 79	OUT (C),A
ED 7A	ADC HL,SP
ED 7B bb aa	LD SP,(aabb)
ED A0	LDI
ED A1	CPI
ED A2	INI
ED A3	OUTI
ED A8	LDD
ED A9	CPD
ED AA	IND
ED AB	OUTD
ED B0	LDIR
ED B1	CPIR
ED B2	INIR
ED B3	OTIR
ED B8	LDDR
ED B9	CPDR
ED BA	INDR
ED BB	OTDR
EE nn	XOR nn
EF	RST 28
F0	RET P
F1	POP AF
F2 bb aa	JP P,aabb
F7	RST 30
F8	RET M
F9	LD SP,HL
FA bb aa	JP M,aabb
FB	EI
FC bb aa	CALL M,aabb
FD 09	ADD IY,BC
FD 19	ADD IY,DE
FD 21 bb aa	LD IY,aabb
FD 22 bb aa	LD (aabb),IY
FD 23	INC IY
FD 29	ADD IY,IY
FD 2A bb aa	LD IY,(aabb)
FD 2B	DEC IY
FD 34 nn	INC (IY+nn)
FD 35 nn	DEC (IY+nn)
FD 36 nn n1	LD (IY+nn),n1
FD 39	ADD IY,SP
FD 46 nn	LD B,(IY+nn)
FD 4E nn	LD C,(IY+nn)
FD 56 nn	LD D,(IY+nn)
FD 5E nn	LD E,(IY+nn)
FD 66 nn	LD H,(IY+nn)
FD 6E nn	LD L,(IY+nn)
FD 70 nn	LD (IY+nn),B
FD 71 nn	LD (IY+nn),C
FD 72 nn	LD (IY+nn),D
FD 73 nn	LD (IY+nn),E
FD 74 nn	LD (IY+nn),H
FD 75 nn	LD (IY+nn),L
FD 77 nn	LD (IY+nn),A

FD 7E nn	LD A,(IY+nn)
FD 86 nn	ADD A,(IY+nn)
FD 8E nn	ADC A,(IY+nn)
FD 96 nn	SUB (IY+nn)
FD 9E nn	SBC A,(IY+nn)
FD A6 nn	AND (IY+nn)
FD AE nn	XOR (IY+nn)
FD B6 nn	OR (IY+nn)
FD CB nn D6	SET 2,(IY+nn)
FD CB nn DE	SET 3,(IY+nn)
FD CB nn E6	SET 4,(IY+nn)
FD CB nn EE	SET 5,(IY+nn)
FD CB nn F6	SET 6,(IY+nn)
FD CB nn FE	SET 7,(IY+nn)
FD E1	POP IY
FD E3	EX (SP),IY

FD BE nn	CP (IY+nn)
FD CB nn 06	RLC (IY+nn)
FD CB nn 0E	RRC (IY+nn)
FD CB nn 16	RL (IY+nn)
FD CB nn 1E	RR (IY+nn)
FD CB nn 26	SLA (IY+nn)
FD CB nn 2E	SRA (IY+nn)
FD CB nn 36	SLI (IY+nn)
FD CB nn 3E	SRL (IY+nn)
FD CB nn 46	BIT 0,(IY+nn)
FD CB nn 4E	BIT 1,(IY+nn)
FD CB nn 56	BIT 2,(IY+nn)
FD CB nn 5E	BIT 3,(IY+nn)
FD CB nn 66	BIT 4,(IY+nn)
FD CB nn 6E	BIT 5,(IY+nn)
FD CB nn 76	BIT 6,(IY+nn)
FD CB nn 7E	BIT 7,(IY+nn)
FD CB nn 86	RES 0,(IY+nn)
FD CB nn 8E	RES 1,(IY+nn)
FD CB nn 96	RES 2,(IY+nn)
FD CB nn 9E	RES 3,(IY+nn)
FD CB nn A6	RES 4,(IY+nn)
FD CB nn AE	RES 5,(IY+nn)
FD CB nn B6	RES 6,(IY+nn)
FD CB nn BE	RES 7,(IY+nn)
FD CB nn C6	SET 0,(IY+nn)
FD CB nn CE	SET 1,(IY+nn)
FD E5	PUSH IY
FD E9	JP (IY)
FD F9	LD SP,IY
FE nn	CP nn
FF	RST 38

Instruction set in Alphabetical order

8E	ADC A,(HL)
DD 8E nn	ADC A,(IX+nn)
FD 8E nn	ADC A,(IY+nn)
8F	ADC A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
CE nn	ADC A,nn
ED 4A	ADC HL,BC
ED 5A	ADC HL,DE
ED 6A	ADC HL,HL
ED 7A	ADC HL,SP
86	ADD A,(HL)
DD 86 nn	ADD A,(IX+nn)
FD 86 nn	ADD A,(IY+nn)
87	ADD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
C6 nn	ADD A,nn
09	ADD HL,BC
19	ADD HL,DE
29	ADD HL,HL
39	ADD HL,SP
DD 09	ADD IX,BC
DD 19	ADD IX,DE
DD 29	ADD IX,IX
CB 49	BIT 1,C
CB 4A	BIT 1,D
CB 4B	BIT 1,E

DD 39	ADD IX,SP
FD 09	ADD IY,BC
FD 19	ADD IY,DE
FD 29	ADD IY,IY
FD 39	ADD IY,SP
A6	AND (HL)
DD A6 nn	AND (IX+nn)
FD A6 nn	AND (IY+nn)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6 nn	AND nn
CB 46	BIT 0,(HL)
DD CB nn 46	BIT 0,(IX+nn)
FD CB nn 46	BIT 0,(IY+nn)
CB 47	BIT 0,A
CB 40	BIT 0,B
CB 41	BIT 0,C
CB 42	BIT 0,D
CB 43	BIT 0,E
CB 44	BIT 0,H
CB 45	BIT 0,L
CB 4E	BIT 1,(HL)
DD CB nn 4E	BIT 1,(IX+nn)
FD CB nn 4E	BIT 1,(IY+nn)
CB 4F	BIT 1,A
CB 48	BIT 1,B
CB 61	BIT 4,C
CB 62	BIT 4,D
CB 63	BIT 4,E

CB 4C	BIT 1,H
CB 4D	BIT 1,L
CB 56	BIT 2,(HL)
DD CB nn 56	BIT 2,(IX+nn)
FD CB nn 56	BIT 2,(IY+nn)
CB 57	BIT 2,A
CB 50	BIT 2,B
CB 51	BIT 2,C
CB 52	BIT 2,D
CB 53	BIT 2,E
CB 54	BIT 2,H
CB 55	BIT 2,L
CB 5E	BIT 3,(HL)
DD CB nn 5E	BIT 3,(IX+nn)
FD CB nn 5E	BIT 3,(IY+nn)
CB 5F	BIT 3,A
CB 58	BIT 3,B
CB 59	BIT 3,C
CB 5A	BIT 3,D
CB 5B	BIT 3,E
CB 5C	BIT 3,H
CB 5D	BIT 3,L
CB 66	BIT 4,(HL)
DD CB nn 66	BIT 4,(IX+nn)
FD CB nn 66	BIT 4,(IY+nn)
CB 67	BIT 4,A
CB 60	BIT 4,B
CB 79	BIT 7,C
CB 7A	BIT 7,D
CB 7B	BIT 7,E
CB 7C	BIT 7,H
CB 7D	BIT 7,L
DC bb aa	CALL C,aabb
FC bb aa	CALL M,aabb
D4 bb aa	CALL NC,aabb
CD bb aa	CALL aabb

CB 64	BIT 4,H
CB 65	BIT 4,L
CB 6E	BIT 5,(HL)
DD CB nn 6E	BIT 5,(IX+nn)
FD CB nn 6E	BIT 5,(IY+nn)
CB 6F	BIT 5,A
CB 68	BIT 5,B
CB 69	BIT 5,C
CB 6A	BIT 5,D
CB 6B	BIT 5,E
CB 6C	BIT 5,H
CB 6D	BIT 5,L
CB 76	BIT 6,(HL)
DD CB nn 76	BIT 6,(IX+nn)
FD CB nn 76	BIT 6,(IY+nn)
CB 77	BIT 6,A
CB 70	BIT 6,B
CB 71	BIT 6,C
CB 72	BIT 6,D
CB 73	BIT 6,E
CB 74	BIT 6,H
CB 75	BIT 6,L
CB 7E	BIT 7,(HL)
DD CB nn 7E	BIT 7,(IX+nn)
FD CB nn 7E	BIT 7,(IY+nn)
CB 7F	BIT 7,A
CB 78	BIT 7,B
2F	CPL
27	DAA
35	DEC (HL)
DD 35 nn	DEC (IX+nn)
FD 35 nn	DEC (IY+nn)
3D	DEC A
05	DEC B
0B	DEC BC

```
C4 bb aa    CALL NZ,aabb
F4 bb aa    CALL P,aabb
EC bb aa    CALL PE,aabb
E4 bb aa    CALL PO,aabb
CC bb aa    CALL Z,aabb

3F          CCF

BE          CP (HL)
DD BE nn    CP (IX+nn)
FD BE nn    CP (IY+nn)
BF          CP A
B8          CP B
B9          CP C
BA          CP D
BB          CP E
BC          CP H
BD          CP L
FE nn       CP nn

ED A9       CPD
ED B9       CPDR
ED A1       CPI
ED B1       CPIR

ED 46       IM 0
ED 56       IM 1
ED 5E       IM 2

ED 78       IN A,(C)
DB nn       IN A,(nn)
ED 40       IN B,(C)
ED 48       IN C,(C)
ED 50       IN D,(C)
ED 58       IN E,(C)
ED 70       IN F,(C)
ED 60       IN H,(C)
ED 68       IN L,(C)

34          INC (HL)
```

```
0D          DEC C
15          DEC D
1B          DEC DE
1D          DEC E
25          DEC H
2B          DEC HL
DD 2B       DEC IX
FD 2B       DEC IY
2D          DEC L
3B          DEC SP

F3          DI

10 nn       DJNZ nn

FB          EI

E3          EX (SP),HL
DD E3       EX (SP),IX
FD E3       EX (SP),IY
08          EX AF,AF'
EB          EX DE,HL
D9          EXX
76          HALT

E9          JP (HL)
DD E9       JP (IX)
FD E9       JP (IY)
DA bb aa    JP C,aabb
FA bb aa    JP M,aabb
D2 bb aa    JP NC,aabb
C3 bb aa    JP aabb
C2 bb aa    JP NZ,aabb
F2 bb aa    JP P,aabb
EA bb aa    JP PE,aabb
E2 bb aa    JP PO,aabb
CA bb aa    JP Z,aabb

38 nn       JR C,nn
18 nn       JR nn
```



```
DD 34 nn    INC (IX+nn)
FD 34 nn    INC (IY+nn)
3C          INC A
04          INC B
03          INC BC
0C          INC C
14          INC D
13          INC DE
1C          INC E
24          INC H
23          INC HL
DD 23       INC IX
FD 23       INC IY
2C          INC L
33          INC SP

ED AA       IND
ED BA       INDR
ED A2       INI
ED B2       INIR

DD 74 nn    LD (IX+nn),H
DD 75 nn    LD (IX+nn),L
DD 36 nn n1 LD (IX+nn),n1

FD 77 nn    LD (IY+nn),A
FD 70 nn    LD (IY+nn),B
FD 71 nn    LD (IY+nn),C
FD 72 nn    LD (IY+nn),D
FD 73 nn    LD (IY+nn),E
FD 74 nn    LD (IY+nn),H
FD 75 nn    LD (IY+nn),L
FD 36 nn n1 LD (IY+nn),n1

32 bb aa    LD (aabb),A
ED 43 bb aa LD (aabb),BC
ED 53 bb aa LD (aabb),DE
22 bb aa    LD (aabb),HL
DD 22 bb aa LD (aabb),IX
FD 22 bb aa LD (aabb),IY
```

```
30 nn       JR NC,nn
20 nn       JR NZ,nn
28 nn       JR Z,nn

02          LD (BC),A
12          LD (DE),A
77          LD (HL),A
70          LD (HL),B
71          LD (HL),C
72          LD (HL),D
73          LD (HL),E
74          LD (HL),H
75          LD (HL),L
36 nn       LD (HL),nn

DD 77 nn    LD (IX+nn),A
DD 70 nn    LD (IX+nn),B
DD 71 nn    LD (IX+nn),C
DD 72 nn    LD (IX+nn),D
DD 73 nn    LD (IX+nn),E

7D          LD A,L
3E nn       LD A,nn
ED 5F       LD A,R

46          LD B,(HL)
DD 46 nn    LD B,(IX+nn)
FD 46 nn    LD B,(IY+nn)
47          LD B,A
40          LD B,B
41          LD B,C
42          LD B,D
43          LD B,E
44          LD B,H
45          LD B,L
06 nn       LD B,nn

ED 4B bb aa LD BC,(aabb)
01 bb aa    LD BC,aabb
```

Opcode	Instruction
DD 66 nn	LD H,(IX+nn)
FD 66 nn	LD H,(IY+nn)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
26 nn	LD H,nn
ED 44	NEG
00	NOP
B6	OR (HL)
DD B6 nn	OR (IX+nn)
FD B6 nn	OR (IY+nn)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
F6 nn	OR nn
ED BB	OTDR
ED B3	OTIR
ED 79	OUT (C),A
ED 41	OUT (C),B
ED 49	OUT (C),C
ED 51	OUT (C),D
ED 59	OUT (C),E
ED 61	OUT (C),H
ED 69	OUT (C),L
D3 nn	OUT (nn),A
ED AB	OUTD

Opcode	Instruction
ED 7B bb aa	LD SP,(aabb)
F9	LD SP,HL
DD F9	LD SP,IX
FD F9	LD SP,IY
31 bb aa	LD SP,aabb
ED A8	LDD
ED B8	LDDR
ED A0	LDI
ED B0	LDIR
DD E1	POP IX
FD E1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DD E5	PUSH IX
FD E5	PUSH IY
CB 86	RES 0,(HL)
DD CB nn 86	RES 0,(IX+nn)
FD CB nn 86	RES 0,(IX+nn)
CB 87	RES 0,A
CB 80	RES 0,B
CB 81	RES 0,C
CB 82	RES 0,D
CB 83	RES 0,E
CB 84	RES 0,H
CB 85	RES 0,L
CB 8E	RES 1,(HL)
DD CB nn 8E	RES 1,(IX+nn)
FD CB nn 8E	RES 1,(IY+nn)
CB 8F	RES 1,A
CB 88	RES 1,B
CB 89	RES 1,C
CB 8A	RES 1,D
CB 8B	RES 1,E

Opcode	Instruction
ED 73 bb aa	LD (aabb),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD 7E nn	LD A,(IX+nn)
FD 7E nn	LD A,(IY+nn)
3A bb aa	LD A,(aabb)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED 57	LD A,I
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
LD D,L	
16 nn	LD D,nn
ED 5B bb aa	LD DE,(aabb)
11 bb aa	LD DE,aabb
5E	LD E,(HL)
DD 5E nn	LD E,(IX+nn)
FD 5E nn	LD E,(IY+nn)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E nn	LD E,nn
66	LD H,(HL)

Opcode	Instruction
4E	LD C,(HL)
DD 4E nn	LD C,(IX+nn)
FD 4E nn	LD C,(IY+nn)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E nn	LD C,nn
56	LD D,(HL)
DD 56 nn	LD D,(IX+nn)
FD 56 nn	LD D,(IY+nn)
2A bb aa	LD HL,(aabb)
21 bb aa	LD HL,aabb
ED 47	LD I,A
DD 21 bb aa	LD IX,aabb
FD 2A bb aa	LD IY,(aabb)
FD 21 bb aa	LD IY,aabb
6E	LD L,(HL)
DD 6E nn	LD L,(IX+nn
FD 6E nn	LD L,(IY+nn)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2E nn	LD L,nn
ED 4F	LD R,A

```
ED A3          OUTI

F1             POP AF
C1             POP BC
D1             POP DE
E1             POP HL

CB 97          RES 2,A
CB 90          RES 2,B
CB 91          RES 2,C
CB 92          RES 2,D
CB 93          RES 2,E
CB 94          RES 2,H
CB 95          RES 2,L

CB 9E          RES 3,(HL)
DD CB nn 9E    RES 3,(IX+nn)
FD CB nn 9E    RES 3,(IY+nn)
CB 9F          RES 3,A
CB 98          RES 3,B
CB 99          RES 3,C
CB 9A          RES 3,D
CB 9B          RES 3,E
CB 9C          RES 3,H
CB 9D          RES 3,L

CB A6          RES 4,(HL)
DD CB nn A6    RES 4,(IX+nn)
FD CB nn A6    RES 4,(IY+nn)
CB A7          RES 4,A
CB A0          RES 4,B
CB A1          RES 4,C
CB A2          RES 4,D
CB A3          RES 4,E
CB A4          RES 4,H
CB A5          RES 4,L

CB AE          RES 5,(HL)
DD CB nn AE    RES 5,(IX+nn)
FD CB nn AE    RES 5,(IY+nn)
```

```
CB 8C          RES 1,H
CB 8D          RES 1,L

CB 96          RES 2,(HL)
DD CB nn 96    RES 2,(IX+nn)
FD CB nn 96    RES 2,(IY+nn)

CB A9          RES 5,C
CB AA          RES 5,D
CB AB          RES 5,E
CB AC          RES 5,H
CB AD          RES 5,L

CB B6          RES 6,(HL)
DD CB nn B6    RES 6,(IX+nn)
FD CB nn B6    RES 6,(IY+nn)
CB B7          RES 6,A
CB B0          RES 6,B
CB B1          RES 6,C
CB B2          RES 6,D
CB B3          RES 6,E
CB B4          RES 6,H
CB B5          RES 6,L

CB BE          RES 7,(HL)
DD CB nn BE    RES 7,(IX+nn)
FD CB nn BE    RES 7,(IY+nn)
CB BF          RES 7,A
CB B8          RES 7,B
CB B9          RES 7,C
CB BA          RES 7,D
CB BB          RES 7,E
CB BC          RES 7,H
CB BD          RES 7,L

C9             RET
D8             RET C
F8             RET M
D0             RET NC
C0             RET NZ
```



```
CB AF          RES 5,A
CB A8          RES 5,B
E0             RET PO
C8             RET Z

ED 4D          RETI
ED 45          RETN

CB 16          RL (HL)
DD CB nn 16    RL (IX+nn)
FD CB nn 16    RL (IY+nn)
CB 17          RL A
CB 10          RL B
CB 11          RL C
CB 12          RL D
CB 13          RL E
CB 14          RL H
CB 15          RL L

17             RLA

CB 06          RLC (HL)
DD CB nn 06    RLC (IX+nn)
FD CB nn 06    RLC (IY+nn)
CB 07          RLC A
CB 00          RLC B
CB 01          RLC C
CB 02          RLC D
CB 03          RLC E
CB 04          RLC H
CB 05          RLC L

07             RLCA

ED 6F          RLD

CB 1E          RR (HL)

9E             SBC A,(HL)
DD 9E nn       SBC A,(IX+nn)
```

```
F0             RET P
E8             RET PE
DD CB nn 1E    RR (IX+nn)
FD CB nn 1E    RR (IY+nn)
CB 1F          RR A
CB 18          RR B
CB 19          RR C
CB 1A          RR D
CB 1B          RR E
CB 1C          RR H
CB 1D          RR L

1F             RRA

CB 0E          RRC (HL)
DD CB nn 0E    RRC (IX+nn)
FD CB nn 0E    RRC (IY+nn)
CB 0F          RRC A
CB 08          RRC B
CB 09          RRC C
CB 0A          RRC D
CB 0B          RRC E
CB 0C          RRC H
CB 0D          RRC L

0F             RRCA

ED 67          RRD

C7             RST 0
CF             RST 8h
D7             RST 10h
DF             RST 18h
E7             RST 20h
EF             RST 28h
F7             RST 30h
FF             RST 38h

CB C9          SET 1,C
CB CA          SET 1,D
```

Opcode	Instruction
FD 9E nn	SBC A,(IY+nn)
9F	SBC A,A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
DE nn	SBC A,nn
ED 42	SBC HL,BC
ED 52	SBC HL,DE
ED 62	SBC HL,HL
ED 72	SBC HL,SP
37	SCF
CB C6	SET 0,(HL)
DD CB nn C6	SET 0,(IX+nn)
FD CB nn C6	SET 0,(IY+nn)
CB C7	SET 0,A
CB C0	SET 0,B
CB C1	SET 0,C
CB C2	SET 0,D
CB C3	SET 0,E
CB C4	SET 0,H
CB C5	SET 0,L
CB CE	SET 1,(HL)
DD CB nn CE	SET 1,(IX+nn)
FD CB nn CE	SET 1,(IY+nn)
CB CF	SET 1,A
CB C8	SET 1,B
CB E3	SET 4,E
CB E4	SET 4,H
CB E5	SET 4,L
CB EE	SET 5,(HL)
DD CB nn EE	SET 5,(IX+nn)

Opcode	Instruction
CB CB	SET 1,E
CB CC	SET 1,H
CB CD	SET 1,L
CB D6	SET 2,(HL)
DD CB nn D6	SET 2,(IX+nn)
FD CB nn D6	SET 2,(IY+nn)
CB D7	SET 2,A
CB D0	SET 2,B
CB D1	SET 2,C
CB D2	SET 2,D
CB D3	SET 2,E
CB D4	SET 2,H
CB D5	SET 2,L
CB DE	SET 3,(HL)
DD CB nn DE	SET 3,(IX+nn)
FD CB nn DE	SET 3,(IY+nn)
CB DF	SET 3,A
CB D8	SET 3,B
CB D9	SET 3,C
CB DA	SET 3,D
CB DB	SET 3,E
CB DC	SET 3,H
CB DD	SET 3,L
CB E6	SET 4,(HL)
DD CB nn E6	SET 4,(IX+nn)
FD CB nn E6	SET 4,(IY+nn)
CB E7	SET 4,A
CB E0	SET 4,B
CB E1	SET 4,C
CB E2	SET 4,D
CB 26	SLA (HL)
DD CB nn 26	SLA (IX+nn)
FD CB nn 26	SLA (IY+nn)
CB 27	SLA A
CB 20	SLA B
CB 21	SLA C

Opcode	Instruction
FD CB nn EE	SET 5,(IY+nn)
CB EF	SET 5,A
CB E8	SET 5,B
CB E9	SET 5,C
CB EA	SET 5,D
CB EB	SET 5,E
CB EC	SET 5,H
CB ED	SET 5,L
CB F6	SET 6,(HL)
DD CB nn F6	SET 6,(IX+nn)
FD CB nn F6	SET 6,(IY+nn)
CB F7	SET 6,A
CB F0	SET 6,B
CB F1	SET 6,C
CB F2	SET 6,D
CB F3	SET 6,E
CB F4	SET 6,H
CB F5	SET 6,L
CB FE	SET 7,(HL)
DD CB nn FE	SET 7,(IX+nn)
FD CB nn FE	SET 7,(IY+nn)
CB FF	SET 7,A
CB F8	SET 7,B
CB F9	SET 7,C
CB FA	SET 7,D
CB FB	SET 7,E
CB FC	SET 7,H
CB FD	SET 7,L
CB 3F	SRL A
CB 38	SRL B
CB 39	SRL C
CB 3A	SRL D
CB 3B	SRL E
CB 3C	SRL H
CB 3D	SRL L
96	SUB (HL)

Opcode	Instruction
CB 22	SLA D
CB 23	SLA E
CB 24	SLA H
CB 25	SLA L
CB 36	SLI (HL)
DD CB nn 36	SLI (IX+nn)
FD CB nn 36	SLI (IY+nn)
CB 37	SLI A
CB 30	SLI B
CB 31	SLI C
CB 32	SLI D
CB 33	SLI E
CB 34	SLI H
CB 35	SLI L
CB 2E	SRA (HL)
DD CB nn 2E	SRA (IX+nn)
FD CB nn 2E	SRA (IY+nn)
CB 2F	SRA A
CB 28	SRA B
CB 29	SRA C
CB 2A	SRA D
CB 2B	SRA E
CB 2C	SRA H
CB 2D	SRA L
CB 3E	SRL (HL)
DD CB nn 3E	SRL (IX+nn)
FD CB nn 3E	SRL (IY+nn)
94	SUB H
95	SUB L
D6 nn	SUB nn
AE	XOR (HL)
DD AE nn	XOR (IX+nn)
FD AE nn	XOR (IY+nn)
AF	XOR A
A8	XOR B

DD 96 nn	SUB (IX+nn)
FD 96 nn	SUB (IY+nn)
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E

A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE nn	XOR nn

HEX	DEC *256	DEC	H	D *256	D	H	D *256	D	H	D *256	D	H	D *256	D
00	00000	0	34	13312	52	68	26624	104	9C	39936	156	D0	53248	208
01	00256	1	35	13568	53	69	26880	105	9D	40192	157	D1	53504	209
02	00512	2	36	13824	54	6A	27136	106	9E	40448	158	D2	53760	210
03	00768	3	37	14080	55	6B	27392	107	9F	40704	159	D3	54016	211
04	01024	4	38	14336	56	6C	27648	108	A0	40960	160	D4	54272	212
05	01280	5	39	14592	57	6D	27904	109	A1	41216	161	D5	54528	213
06	01536	6	3A	14848	58	6E	28160	110	A2	41472	162	D6	54784	214
07	01792	7	3B	15104	59	6F	28416	111	A3	41728	163	D7	55040	215
08	02048	8	3C	15360	60	70	28672	112	A4	41984	164	D8	55296	216
09	02304	9	3D	15616	61	71	28928	113	A5	42240	165	D9	55552	217
0A	02560	10	3E	15872	62	72	29184	114	A6	42496	166	DA	55808	218
0B	02816	11	3F	16128	63	73	29440	115	A7	42752	167	DB	56064	219
0C	03072	12	40	16384	64	74	29696	116	A8	43008	168	DC	56320	220
0D	03328	13	41	16640	65	75	29952	117	A9	43264	169	DD	56576	221
0E	03584	14	42	16896	66	76	30208	118	AA	43520	170	DE	56832	222
0F	03840	15	43	17152	67	77	30464	119	AB	43776	171	DF	57088	223
10	04096	16	44	17408	68	78	30720	120	AC	44032	172	E0	57344	224
11	04352	17	45	17664	69	79	30976	121	AD	44288	173	E1	57600	225
12	04608	18	46	17920	70	7A	31232	122	AE	44544	174	E2	57856	226
13	04864	19	47	18176	71	7B	31488	123	AF	44800	175	E3	58112	227
14	05120	20	48	18432	72	7C	31744	124	B0	45056	176	E4	58368	228
15	05376	21	49	18688	73	7D	32000	125	B1	45312	177	E5	58624	229
16	05632	22	4A	18944	74	7E	32256	126	B2	45568	178	E6	58880	230
17	05888	23	4B	19200	75	7F	32512	127	B3	45824	179	E7	59136	231
18	06144	24	4C	19456	76	80	32768	128	B4	46080	180	E8	59392	232
19	06400	25	4D	19712	77	81	33024	129	B5	46336	181	E9	59648	233
1A	06656	26	4E	19968	78	82	33280	130	B6	46592	182	EA	59904	234
1B	06912	27	4F	20224	79	83	33536	131	B7	46848	183	EB	60160	235
1C	07168	28	50	20480	80	84	33792	132	B8	47104	184	EC	60416	236
1D	07424	29	51	20736	81	85	34048	133	B9	47360	185	ED	60672	237
1E	07680	30	52	20992	82	86	34304	134	BA	47616	186	EE	60928	238
1F	07936	31	53	21248	83	87	34560	135	BB	47872	187	EF	61184	239
20	08192	32	54	21504	84	88	34816	136	BC	48128	188	F0	61440	240
21	08448	33	55	21760	85	89	35072	137	BD	48384	189	F1	61696	241
22	08704	34	56	22016	86	8A	35328	138	BE	48640	190	F2	61952	242
23	08960	35	57	22272	87	8B	35584	139	BF	48896	191	F3	62208	243
24	09216	36	58	22528	88	8C	35840	140	C0	49152	192	F4	62464	244
25	09472	37	59	22784	89	8D	36096	141	C1	49408	193	F5	62720	245
26	09728	38	5A	23040	90	8E	36352	142	C2	49664	194	F6	62976	246
27	09984	39	5B	23296	91	8F	36608	143	C3	49920	195	F7	63232	247
28	10240	40	5C	23552	92	90	36864	144	C4	50176	196	F8	63488	248
29	10496	41	5D	23808	93	91	37120	145	C5	50432	197	F9	63744	249
2A	10752	42	5E	24064	94	92	37376	146	C6	50688	198	FA	64000	250
2B	11008	43	5F	24320	95	93	37632	147	C7	50944	199	FB	64256	251
2C	11264	44	60	24576	96	94	37888	148	C8	51200	200	FC	64512	252
2D	11520	45	61	24832	97	95	38144	149	C9	51456	201	FD	64768	253
2E	11776	46	62	25088	98	96	38400	150	CA	51712	202	FE	65024	254
2F	12032	47	63	25344	99	97	38656	151	CB	51968	203	FF	65280	255
30	12288	48	64	25600	100	98	38912	152	CC	52224	204			
31	12544	49	65	25856	101	99	39168	153	CD	52480	205			
32	12800	50	66	26112	102	9A	39424	154	CE	52736	206			
33	13056	51	67	26368	103	9B	39680	155	CF	52992	207			

The left column is the Hex code.
The centre column is the decimal equivalent multiplied by 256 for calculating the M.S.B
The third column is for use with the L.S.B. or single byte

هذا الكتاب

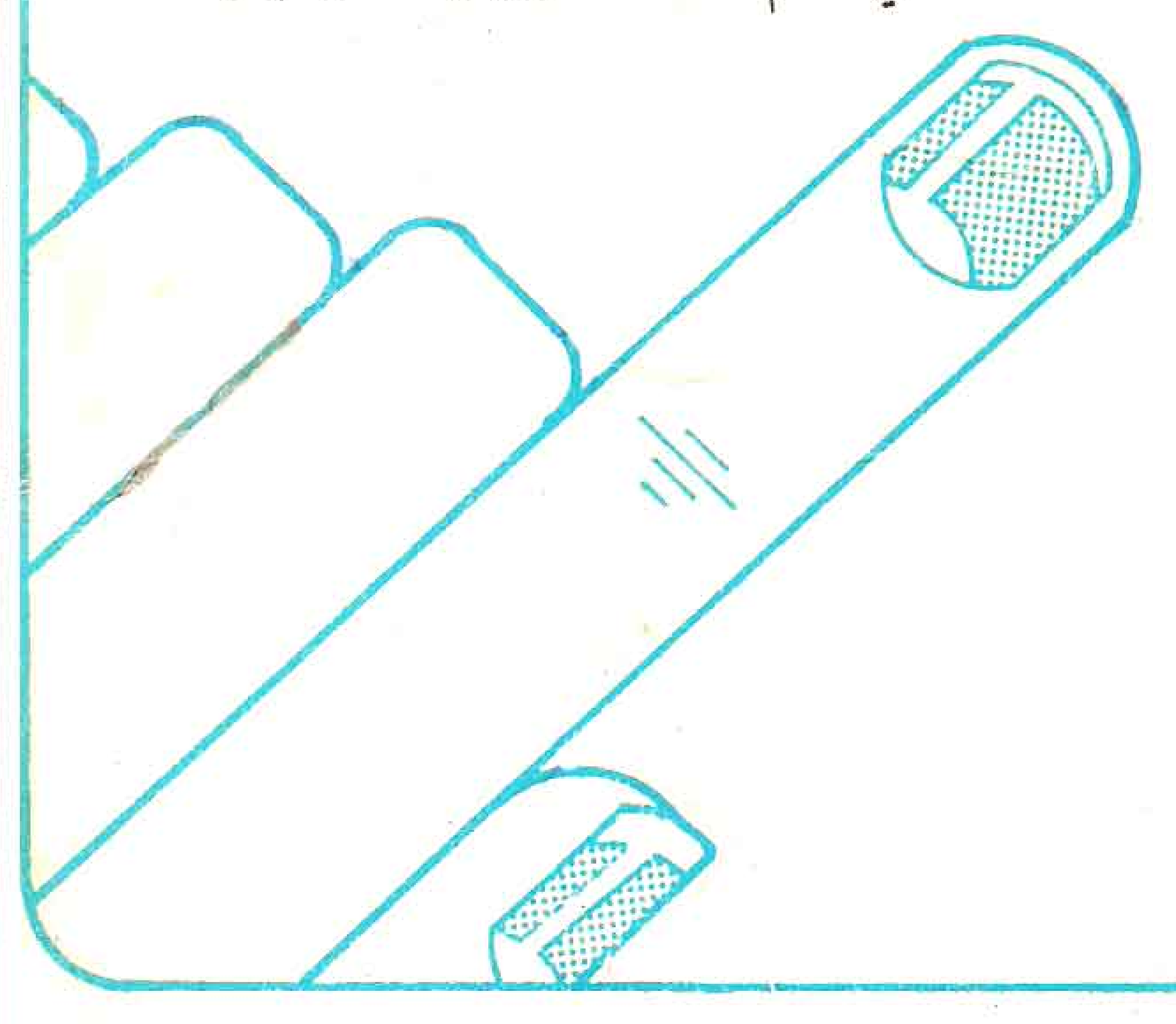
● «البرمجة بلغة الآلة لنظام MSX» هو الكتاب الثاني في «سلسلة نظام MSX» وهي سلسلة من الكتب تقوم بتغطية جميع جوانب هذا النظام القياسي، وتقدم مادة ممتازة لاغنى عنها، تفيد كل مستخدم لنظام MSX.

● «البرمجة بلغة الآلة لنظام MSX» هو الدليل الدقيق لجميع الحاسبات الالكترونية التي تحمل علامة MSX ومهما كانت تسميتها التجارية.

● إنه كتاب قيّم لمستخدمي نظام الـ MSX وبشكل خاص للمتقدمين منهم، لأنه يساعدهم على التعامل مع أجهزتهم وكتابة وتنفيذ برامجهم، بسرعة كبيرة مقارنة مع اللغات الأخرى.

● يقوم هذا الكتاب بمعالجة النقاط التالية:

- التعريف بلغة الآلة بشكل عام.
- شرح تعليمات المعالج Z80 و Z80A وطرق استخدامها.
- استخدام المجمع ZEN لكتابة برامج بلغة الآلة.
- شرح الروتينات المستخدمة في نظام الـ MSX وكيفية معالجتها.



دار الحضارة
للنشر والتوزيع